30th Annual Conference on Neural Information Processing Systems (NIPS 2016)

Pre-Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches (CoCo @ NIPS 2016)

Tarek R. Besold, Antoine Bordes, Artur D'Avila Garcez, and Greg Wayne
(eds.)

Barcelona, Catalonia, Spain, 9th of December 2016

Invited Speakers of CoCo @ NIPS 2016

- Barbara Hammer, Bielefeld University.
- Pascal Hitzler, Wright State University.
- Risto Miikkulainen, University of Texas at Austin & Sentient Technologies, Inc.
- Dan Roth, University of Illinois at Urbana-Champaign.
- Kristina Toutanova, Microsoft Research.

Panelists on "Explainable AI" of CoCo @ NIPS 2016

- Yoshua Bengio, University of Montreal.
- Marco Gori, University of Siena.
- Alessio Lomuscio, Imperial College London.
- Gary Marcus, New York University & Geometric Intelligence Inc.
- Stephen H. Muggleton, Imperial College London.
- Michael Witbrock, IBM Research.

Contents: Contributed Papers

Neuro-Symbolic EDA-based Optimisation using ILP-enhanced DBNs
(*Saikia, Vig, Srinivasan, Shroff, Agarwal & Richa*)

Top-Down and Bottom-Up Interactions between Low-Level Reactive Control and Symbolic Rule Learning Embodied Agents
(*Moulin-Frier, Arsiwalla, Puigbo, Sanchez-Fibla, Duff & Verschure*)

MS MARCO: A Human-Generated Machine Reading Comprehension Dataset
(*Nguyen, Rosenberg, Song, Gao, Tiwary, Majumder & Deng*)

Accuracy and Interpretability Trade-offs in Machine Learning Applied to Safer Gambling
(*Weyde & D'Avila Garcez*)

Variable binding through assemblies in spiking neural networks

Robert Legenstein
Institute for Theoretical Computer Science
Graz University of Technology
A-8010 Graz, Austria
legi@igi.tugraz.at

Christos H. Papadimitriou
EECS
UC Berkeley
CA 94720, USA
christos@cs.berkeley.edu

Santosh Vempala
College of Computing
Georgia Tech
Atlanta, GA 30308, USA
vempala@gatech.edu

Wolfgang Maass
Institute for Theoretical Computer Science
Graz University of Technology
A-8010 Graz, Austria
maass@igi.tugraz.at

Abstract

We propose a model for the binding of variables to concrete fillers in the human brain. The model is based on recent experimental data about corresponding neural processes in humans. First, electrode recordings from the human brain suggest that concepts are represented in the medial temporal lobe (MTL) through sparse sets of neurons (assemblies). Second, fMRI recordings from the human brain suggest that specific subregions of the temporal cortex are dedicated to the representation of specific roles (e.g., subject or object) of concepts in a sentence or visually presented episode. We propose that quickly recruited assemblies of neurons in these subregions act as pointers to previously created assemblies that represent concepts. As a proof of principle, we performed computer simulations of a spiking neural network model that implemented the proposed paradigm for binding through assembly pointers. We show that the model supports basic operations of brain computations, such as structured recall and copying of information.

1 Introduction

Numerous electrode recordings from the human brain (see [1] for a review) suggest that concepts are represented through sparse sets of neurons that fire (more or less) whenever the corresponding concept is activated. These data confirm earlier hypotheses and models about the representation of tokens of cognitive computations through assemblies of neurons [2]. More recent data [3] suggests that assemblies should not be seen as invariant entities, but as fluent coalitions of neurons whose synaptic interconnections can be strengthened very fast, even in response to a single experience. This data also suggests that these processes on the synaptic level underlie the formation of associations.

We propose that assemblies of neurons are also instrumental for creating a transient or longer lasting binding of a variable to a filler. For example, they could bind a variable that represents a thematic role (e.g., agent or patient in an episode) to a word or concept. Information about the neural representation of semantic roles is provided through recent fMRI data, where specific subregions in the temporal cortex were shown to respond to specific semantic (thematic) roles of individuals in an episode that was communicated through a sentence [4] or a movie [5].

Here we do not assume that semantic roles are represented by fixed assemblies of neurons. Such a fixed assembly would in general not have sufficient direct synaptic connectivity to the virtually unlimited repertoire of words or concepts, each represented through assemblies in other brain regions, that could acquire this semantic role in an episode. To achieve such large potential connectivity, the size of this fixed assembly would have to be so large that its activation would not be consistent with generic sparse firing activity in each brain region. Rather, we propose that the specific subregions of the temporal cortex that were shown to be activated differentially in dependence of the specific semantic role of a concept serve as large pools of neurons (we will refer to them as neural spaces). In neural spaces, sparse assemblies can quickly be recruited from the subset of neurons that happen to have direct synaptic connections to the assemblies for the corresponding concepts involved (assembly pointers). We propose that this model can reconcile functional needs, such as being able to recall the concept from its recent thematic role, with data on the inherently sparse connectivity between brain areas [6]. One can also view this model as a direct extrapolation of data on the formation of associations between concepts from [3] to associations between thematic roles (i.e., variables) and concepts.

We propose that one well-known neurophysiological mechanism is essential for the control of this binding process: disinhibition. At least two different ways how brain areas can be selectively disinhibited have been proposed on the basis of experimental data [7]. One is neuromodulatory control (especially cholinergic), see [8]. Another one is disinhibition via the activation of VIP cells, i.e., of inhibitory neurons that primarily target other types of inhibitory neurons [9]. Firing of VIP cells is apparently often caused by top-down inputs (they are especially frequent in layer 1, where top-down and lateral distal inputs arrive). Their activation is conjectured to enable neural firing and plasticity within specific patches of the brain through disinhibition, see e.g. [7, 8, 10, 11, 12]. We propose that disinhibition plays a central role for neural computation and learning by controlling operations on assembly pointers.

In this article, we briefly describe the proposed model of assembly pointers for variable binding and outline a spiking neural network that implements this model. A more detailed discussion for the model can be found in [13].

2 Results

Recent experimental data indicates that neural activity patterns in cortex can be characterized in first approximation as spontaneous and stimulus-evoked switching between the activations of different (but somewhat overlapping) subsets of neurons (see e.g. [14, 15, 16]), often referred to as assemblies of neurons. We therefore represent a specific content (a word or a concept) in our model by a specific assembly of neurons in a content space $\mathcal{C}$.

Our model for the binding of a variable that represents a syntactic role (agent, verb, patient) to a concrete word (referred to more abstractly as "content" in our model) is based on the results and hypotheses of [4]. We refer to the particular region or set of neurons that is reserved for a variable v as a neural space $\mathcal{N}_v$ for variable v. Thus each such neural space $\mathcal{N}_v$ can be viewed as functioning like a register in a computer in the terminology of [4]. But in contrast to a computer, this "register" is not used for storing content in it. Rather, assemblies in this register store "handles" or "pointers" to assemblies that store content information in the separate content space $\mathcal{C}$.

In addition our model takes into account that neurons typically do not fire just because they receive sufficiently strong excitatory input. Experimental data suggest that neurons are typically prevented from firing by an "inhibitory lock", that balances or even dominates excitatory input [17]. Thus a generic pyramidal cell is likely to fire because two events take place: its inhibitory lock is temporarily lifted ("disinhibition") and its excitatory input is sufficiently strong. Such disinhibition is apparently often caused by top-down inputs. We propose that orchestrated top-down disinhibition of neural spaces controls the formation of assembly pointers as well as the recall of content from assembly pointers and other cognitive operations.

We implemented this network structure with stochastically spiking neurons. The network consisted of a content space $\mathcal{C}$ and a neural space $\mathcal{N}_v$ for some variable v that each contained 1000 recurrently connected excitatory neurons (connection probability 0.1). To ensure sparse activity, lateral inhibition was implemented in a symbolic manner in each of the neural spaces through an inhibitory current that depended on the recent firing rate of neurons in the space. Disinhibition was modeled

through a multiplicative effect of an inhibitory input on the membrane potential of neurons. Reciprocal connections between $\mathcal{C}$ and $\mathcal{N}_v$ were introduced randomly with a connection probability of 0.1. Neurons in the content space received in addition connections from 200 input neurons. All synapses between excitatory neurons in the circuit were subject to spike-timing dependent plasticity (STDP).

First, five assemblies were induced in content space $\mathcal{C}$ by repeated presentation of 5 simple rate patterns $P_1, \ldots, P_5$ that represented concepts or words at the input. Due to these pattern presentations, an assembly $\mathcal{C}(P_i)$ emerged in content space for each of the patterns P_i (assembly sizes between 81 and 86 neurons) that showed robust firing activity whenever the corresponding pattern was presented as input. STDP of recurrent connections led to a strengthening of these synapses within each assembly, while synapses between assemblies remained weak (see [13] for details).

According to our model for variable binding, disinhibition enables the creation of an assembly pointer in some neural space $\mathcal{N}_v$ to the currently active assembly in the content space. Such disinhibition of a neural space $\mathcal{N}_v$ allows that some of neurons in it can fire, especially those that receive sufficiently strong excitatory input from a currently active assembly in the content space. Furthermore, in line with previously cited experimental reports we assume that this allowed firing of neurons in the neural space also enables plasticity of these neurons and synapses that are connected to it. To validate this hypothesis, we simulated disinhibition of the neural space $\mathcal{N}_v$ while input to content space $\mathcal{C}$ excited an assembly there. We found that STDP in the synapses that connect the content space $\mathcal{C}$ and the neural space $\mathcal{N}_v$ led to the stable emergence of an assembly in $\mathcal{N}_v$ within a second. Further, plasticity at recurrent synapses in $\mathcal{N}_v$ induced a strengthening of recurrent connections within assemblies there. Hence, disinhibition led to the rapid and stable creation of an assembly in the neural space, i.e., an assembly pointer. We denote such creation of an assembly pointer in a neural space $\mathcal{N}_v$ for a specific variable v to content P encoded in content space by CREATE(v, P).

Our model for variable binding based on assembly pointers further assumes that strengthened synaptic connections between assemblies in a neural space $\mathcal{N}_v$ and content space $\mathcal{C}$ enable the recall RECALL(v) of the variables' content, i.e., the activation of the assembly for content P in content space that was active at the most recent CREATE(v, P) operation (e.g., representing the word "truck"). It has been shown that the excitability of pyramidal cells can be changed in a very fast but transient manner through fast depression of GABA-ergic synapses onto pyramidal cells [18]. Using such a mechanism, a RECALL(v) can be initiated by disinhibition of the neural space $\mathcal{N}_v$ while the content space does not receive any bottom up input. The increased excitability of recently activated neurons in $\mathcal{N}_v$ ensures that the most recently active assembly is activated which in turn activates the corresponding content through its (previously potentiated) feedback connections to content space $\mathcal{C}$. The viability of this model for the recall of previously bound content was confirmed in simulations of the spiking neural network model described above, see [13] for details.

Apart from the creation of assembly pointers and recall of content, two further operations have been postulated to be essential for many higher cognitive functions [19]. The first is COPY(u, v) that copies the content of variable u to variable v. In our model, the copy operation creates an assembly pointer in neural space $\mathcal{N}_v$ for variable v to the content to which the assembly pointer in neural space $\mathcal{N}_u$ for variable u refers to. This operation can be implemented in our model simply by disinhibiting $\mathcal{N}_u$ in order to activate the corresponding content in $\mathcal{C}$ followed by a disinhibition of $\mathcal{N}_v$ in order to create an assembly pointer there. A final fundamental operation considered in [19] is COMPARE(u, v) which compares whether the content of u equals the content of v. One possible implementation of this operation in our model is a readout neuron that receives depressing synaptic connections from the content space. Then, when the content for $\mathcal{N}_u$ and $\mathcal{N}_v$ is recalled in sequence, readout synapses will be depressed for the content of $\mathcal{N}_v$ if and only if the content of $\mathcal{N}_u$ equals the content of $\mathcal{N}_v$. Such a "change detecting" readout thus exhibits high activity if the contents of $\mathcal{N}_u$ and $\mathcal{N}_v$ are different. We confirmed in computer simulations that these operations can be implemented in a spiking neural network model of assembly pointers, see [13].

Reproducing experimental data on the binding of agents to roles: Two experiments were performed in [4] that provided new insights in how variables may be encoded in cortex. Sentences were shown to participants where individual words (like "truck" or "ball") can occur as the agent or as the patient. In a first experiment, the authors aimed to identify cortical regions that encode the meaning of such sentences. Four example sentences with the words "truck" and "ball" are "The truck hit the ball" (S1), "The ball was hit by the truck" (S2), "The truck was hit by the ball" (S3), and "The ball hit the truck" (S4). Here, S1 and S2 (and S3 and S4 respectively) have the same meaning. Indeed,

the authors showed that a linear classifier is able to classify the meaning of such sentences from the fMRI signal of left mid-superior temporal cortex (lmSTC). Using our model for assembly pointers, we can model such situations by binding words either to an agent variable ("who did it") or to a patient variable ("to whom it was done"). Under the assumption that lmSTC hosts neural spaces (with assembly pointers) for the role of words, it is expected that the meaning of a sentence can be decoded from the activity there, but not from the activity in content space where the identities are encoded independently of their role. This conjecture was verified through computer simulations of our spiking neural network model for assembly pointers, see [13] for details.

A second experiment in [4] revealed that subregions of lmSTC also contain information about the current value of the variables for the agent and the patient. More specifically, the authors showed that one is able to predict from the fMRI signal of one subregion of lmSTC the identity of the agent and from the signal in another subregion the identity of the patient (generalizing over all identities of other roles and over different verbs). We confirmed through computer simulations that this is also the case in the proposed model since the assemblies that are formed in the neural spaces $\mathcal{N}_{\text{agent}}$ and $\mathcal{N}_{\text{patient}}$ are typically specific to the bound content. Note that such classification would fail if each neural space consisted of only a single assembly that is activated for all possible fillers [19], since in this case no information about the identity of the role is available in the neural space for the variable.

3 Discussion

It has often been emphasized (see e.g. [20, 21]) that there is a need to understand brain mechanisms for variable binding, and several models for variable binding had been proposed in the literature. These models fall into one of the general classes of pointer-based binding, binding by synchrony, or convolutional binding. Pointer-based models (e.g., [19, 22]) assume that pointers are implemented by single neurons or populations of neurons which are activated as a whole group. In contrast, our model is based on the assumption that distributed assemblies of neurons are the fundamental tokens for encoding symbols and content in the brain, and also for pointers. We propose that these assembly pointers can be created on the fly in some neural spaces for variables and occupy only a sparse subset of neurons in these spaces. It has been shown in [4] that the filler of a thematic role (e.g. the actor) can be predicted from the fMRI signal of a subregion in temporal cortex when a person reads a sentence. As shown above, this finding is consistent with assembly pointers. It is however inconsistent with models where a variable engages a population of neurons that is independent of the bound content, such as traditional pointer-based models. In comparison to traditional pointer models, the assembly pointer model could also give rise to a number of functional advantages. In a neural space $\mathcal{N}_v$ for a variable v, several instantiations of the variable can coexist at the same time, since they can be represented there by increased excitabilities of different assemblies. These contents could be recalled as different possibilities in a structured recall and combined in content space $\mathcal{C}$ with the content of other variables to in order to answer more complex questions.

Some data shows that the relation between spiking activity and the phases of underlying oscillatory population activity may play a role in hippocampus and for working memory [23], indicating a possible role of synchrony in the binding process. Still, the reliability and capacity of binding by synchrony is currently unclear. We note that, while our model is not based on precise synchronization of spikes in different neural spaces, the synaptic coupling between these spaces together with lateral inhibition leads to some synchronized oscillations of interacting neural spaces in our simulations. This is consistent with recent experimental data which suggest that common rhythms in two brain areas support the flow of excitation between these two areas, and also the potentiation of synapses between activated neurons in both areas [24].

Convolutional binding (see e.g., [25]) uses mathematical operations on high-dimensional vectors for variable binding. It had been used in the semantic pointer architecture of Eliasmith [26] where spiking neural networks were constructed to perform these rather complex operations. Similarly, the neural blackboard architecture (NBA, see e.g. [27]) relies on a number of neural circuits that were constructed for example to gate activity or to memorize associations. In contrast to these models, the assembly pointer model focuses on the emergence of binding operations, using assumptions on the fundamental level of assembly coding, network connectivity statistics, and plasticity processes.

We have presented in this article a model for variable binding through assemblies based on "assembly pointers". The model is consistent with recent findings on cortical assemblies and the encoding of

sentence meaning in cortex [4]. It provides a direct link between information processing on the level of symbols and sentences and processes on the level of neurons and synapses. The resulting model for brain computation allows top down structuring of incoming information, thereby laying the foundation of goal oriented „willful" information processing rather than just input-driven processing.

Acknowledgments

Written under partial support by the European Union project #604102 (Human Brain Project).

References

[1] R. Q. Quiroga. Neuronal codes for visual perception and memory. *Neuropsychologia*, 83:227–241, 2016.

[2] D. O. Hebb. *The Organization of Behavior*. Wiley, New York, 1949.

[3] M. J. Ison, R. Q. Quiroga, and I. Fried. Rapid encoding of new memories by individual neurons in the human brain. *Neuron*, 87(1):220–230, 2015.

[4] S. M. Frankland and J. D. Greene. An architecture for encoding sentence meaning in left mid-superior temporal cortex. *Proceedings of the National Academy of Sciences*, 112(37):11732–11737, 2015.

[5] J. Wang, V. L. Cherkassky, Y. Yang, K. K., Chang, R. Vargas, N. Diana, and M.A. Just. Identifying thematic roles from neural representations measured by functional magnetic resonance imaging. *Cognitive Neuropsychology*, 33(3-4):257–264, 2016.

[6] X. J. Wang and H. Kennedy. Brain structure and dynamics across scales: in search of rules. *Current opinion in neurobiology*, 37:92–98, 2016.

[7] J. J. Letzkus, S. B. E. Wolff, and A. Lüthi. Disinhibition, a circuit mechanism for associative learning and memory. *Neuron*, 88:264–276, 2015.

[8] R. C. Froemke and C. E. Schreiner. Synaptic plasticity as a cortical coding scheme. *Current Opinion in Neurobiology*, 35:185–199, 2015.

[9] K. D. Harris and G. M. G. Shepherd. The neocortical circuit: themes and variations. *Nature Neuroscience*, 18(2):170–181, 2015.

[10] P. Caroni. Inhibitory microcircuit modules in hippocampal learning. *Current Opinion in Neurobiology*, 35:66–73, 2015.

[11] C. K. Pfeffer. Inhibitory neurons: VIP cells hit the brake on inhibition. *Current Biology*, 24(1):R18–R20, 2014.

[12] Y. Fu, M. Kaneko, Y. Tang, a. Alvarez-Buylla, and M. P. Stryker. A cortical disinhibitory circuit for enhancing adult plasticity. *Elife*, 4:e05558, 2015.

[13] R. Legenstein, C. H. Papadimitriou, S. Vempala, and W. Maass. Assembly pointers for variable binding in networks of spiking neurons. *arXiv:1611.03698*, 2016.

[14] G. Buzsaki. Neural syntax: cell assemblies, synapsembles, and readers. *Neuron*, 68(3):362–385, 2010.

[15] B. Bathellier, L. Ushakova, and S. Rumpel. Discrete neocortical dynamics predict behavioral categorization of sounds. *Neuron*, 76(2):435–449, 2012.

[16] A. Luczak and J. N. MacLean. Default activity patterns at the neocortical microcircuit level. *Frontiers in Integrative Neuroscience*, 6(30):doi: 10.3389/fnint.2012.00030, 2012.

[17] B. Haider, M. Häusser, and M. Carandini. Inhibition dominates sensory responses in the awake cortex. *Nature*, 493(7430):97–100, 2013.

[18] D. M. Kullmann, A. W. Moreau, Y. Bakiri, and E. Nicholson. Plasticity of inhibition. *Neuron*, 75(6):951–962, 2012.

[19] A. D. Zylberberg, L. Paz, P. R. Roelfsema, S. Dehaene, and M. Sigman. A neuronal device for the control of multi-step computations. *Papers in Physics*, 5:050006, 2013.

[20] G. F. Marcus. The algebraic mind: Integrating connectionism and cognitive science. *MIT Press*, 2003.

[21] G. F. Marcus, A. Marblestone, and T. Dean. The atoms of neural computation - does the brain depend on a set of elementary, reusable computations? *Science*, 346(6209):551–552, 2014.

[22] T. Kriete, D. C. Noelle, J. D. Cohen, and R. C. O'Reilly. Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences*, 110(41):16390–16395, 2013.

[23] M. Siegel, M. R. Warden, and E. K. Miller. Phase-dependent neuronal coding of objects in short-term memory. *Proceedings of the National Academy of Sciences*, 106(50):21341–21346, 2009.

[24] A. D. Friederici and W. Singer. Grounding language processing on basic neurophysiological principles. *Trends in Cognitive Sciences*, 19(6):329–338, 2015.

[25] T. A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, 1995.

[26] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen. A large-scale model of the functioning brain. *science*, 338(6111):1202–1205, 2012.

[27] F. Van der Velde and M. De Kamps. Neural blackboard architectures of combinatorial structures in cognition. *Behavioral and Brain Sciences*, 29(01):37–70, 2006.

ReasoNet: Learning to Stop Reading in Machine Comprehension

Yelong Shen, Po-Sen Huang, Jianfeng Gao, Weizhu Chen
Microsoft Research, Redmond, WA, USA
{yeshen,pshuang,jfgao,wzchen}@microsoft.com

Abstract

Teaching a computer to read a document and answer general questions pertaining to the document is a challenging yet unsolved problem. In this paper, we describe a novel neural network architecture called the Reasoning Network (ReasoNet) for machine comprehension tasks. ReasoNets make use of multiple turns to effectively exploit and then reason over the relation among queries, documents, and answers. Different from previous approaches using a fixed number of turns during inference, ReasoNets introduce a termination state to relax this constraint on the reasoning depth. With the use of reinforcement learning, ReasoNets can dynamically determine whether to continue the comprehension process after digesting intermediate results, or to terminate reading when it concludes that existing information is adequate to produce an answer. ReasoNets have achieved state-of-the-art performance in machine comprehension datasets, including unstructured CNN and Daily Mail datasets, and a structured Graph Reachability dataset.

1 Introduction

Teaching machines to read, process, and comprehend natural language documents is a coveted goal for artificial intelligence [2, 17, 6]. Genuine reading comprehension is extremely challenging, since effective comprehension involves thorough understanding of documents and performing sophisticated inference. Toward solving this machine reading comprehension problem, in recent years, several work has collected various datasets, in the form of question, passage, and answer, to test machine on answering a question based on the provided passage [17, 6, 7, 16]. Some large-scale cloze-style datasets [6, 7] have gained significant attention along with powerful deep learning models.

Recent approaches on cloze-style datasets can be separated into two categories: single-turn and multi-turn reasoning. Single turn reasoning models utilize attention mechanisms [1] with deep learning models to emphasize specific parts of the document which are relevant to the query. These attention models subsequently calculate the relevance between a query and the corresponding weighted representations of document subunits (e.g. sentences or words) to score target candidates [7, 6, 8]. However, considering the sophistication of the problem, after a single-turn comprehension, readers often revisit some specific passage or the question to grasp a better understanding of the problem. With this motivation, recent advances in reading comprehension have made use of multiple turns to infer the relation between query, document and answer [7, 5, 22, 18]. By repeatedly processing the document and question after digesting intermediate information, multi-turn reasoning can generally produce a better answer and all existing work has demonstrated its superior performance consistently.

Existing multi-turn models have a fixed number of hops or iterations in their inference, i.e., with predetermined reasoning depth, without regard to the complexity of each individual query or document. However, when a human reads a document with a question in mind, we often decide whether we want to stop reading if we believe the observed information is adequate already to answer the question, or continue reading after digesting intermediate information until we can answer the question with confidence. This behavior generally varies from document to document, or question to question

because it is related to the sophistication of the document or the difficulty of the question. Meanwhile, the analysis in [3] also illustrates the huge variations in the difficulty level with respect to questions in the CNN/Daily Mail datasets [6]. For a significant part of the datasets, this analysis shows that the problem cannot be solved without appropriate reasoning on both its query and document.

With this motivation, we propose a novel neural network architecture called Reasoning Network (ReasoNet). ReasoNets try to mimic the inference process of human readers. With a question in mind, ReasoNets read a document repeatedly, each time focusing on different parts of the document until a satisfying answer is found or formed. This reminds us of a Chinese proverb: *"The meaning of a book will become clear if you read it hundreds of times."*. Moreover, unlike previous approaches using fixed number of hops or iterations, ReasoNets introduce a termination state in the inference. This state can decide whether to continue the inference to next turn after digesting intermediate information, or to terminate the whole inference when it concludes that existing information is sufficient to yield an answer. This number of turns in the inference is dynamically modeled by both the document and the query, and can be learned automatically according to the difficulty of the problem.

One of the significant challenges ReasoNets face is how to design an efficient training method, since the termination state is discrete and not connected to the final output. This prohibits canonical back-propagation method being directly applied to train ReasoNets. Inspired by [24, 13], we tackle this challenge by proposing a novel deep reinforcement learning method called Contrastive Reward (CR) to successfully train ReasoNets. Unlike traditional reinforcement learning optimization methods using a global variable to capture rewards, CR utilizes an instance-based reward baseline assignment. Experiments show the superiority of CR in both training speed and accuracy. Finally, by accounting for a dynamic termination state during inference and applying proposed deep reinforcement learning optimization method, ReasoNets achieve the state-of-the-art results in machine comprehension datasets when the paper is first publicly available in arXiv[1], including unstructured CNN and Daily Mail datasets, and a proposed structured Graph Reachability dataset.

This paper is organized as follows. In Section 2, we review and compare recent work on machine reading comprehension tasks. In Section 3, we introduce our proposed ReasoNet model architecture and training objectives. Section 4 presents the experimental setting and results on unstructured and structured machine reading comprehension tasks .

2 Related Work

Recently, with large-scale datasets available and the impressive advance of various statistical models, machine reading comprehension tasks have attracted much attention. Here we mainly focus on the related work in cloze-style datasets [6, 7]. Based on how they perform the inference, we can classify their models into two categories: single-turn and multi-turn reasoning.

Single-turn reasoning Single turn reasoning models utilize an attention mechanism to emphasis some sections of a document which are relevant to a query. This can be thought of as treating some parts unimportant while focusing on other important ones to find the most probable answer. [6] propose the attentive reader and the impatient reader models using neural networks with an attention over passages to predict candidates. [7] use attention over window-based memory, which encodes a window of words around entity candidates, by leveraging an end-to-end memory network [19]. Meanwhile, given the same entity candidate can appear multiple times in a passage, [8] propose the attention-sum reader to sum up all the attention scores for the same entity. This score captures the relevance between a query and a candidate. [3] propose using a bilinear term similarity function to calculate attention scores with pretrained word embedding. [22] propose the EpiReader which uses two neural network structures: one extracts candidates using the attention-sum reader; the other reranks candidates based on a bilinear term similarity score calculated from query and passage representations.

Multi-turn reasoning For complex passages and complex queries, human readers often revisit the given document in order to perform deeper inference after reading a document. Several recent studies try to simulate this revisit by combining the information in the query with the new information digested from previous iterations [7, 5, 18, 23, 12]. [7] use multiple hops memory network to augment the query with new information from the previous hop. Gated Attention reader [5] is an extension of

[1] https://arxiv.org/abs/1609.05284

Algorithm 1: Stochastic Inference in a ReasoNet

Input : Memory M; Initial state s_1; Step $t = 1$; Maximum Step T_{max}
Output : Termination Step T, Answer a_T

Sample t_t from the distribution $p(\cdot|f_{tg}(s_t; \theta_{tg}))$;
if t_t is false, go to Step 3; otherwise Step 6;
Generate attention vector $x_t = f_{att}(s_t, M; \theta_x)$;
Update internal state $s_{t+1} = \text{RNN}(s_t, x_t; \theta_s)$;
Set $t = t + 1$; if $t < T_{\text{max}}$ go to Step 1; otherwise Step 6;
Generate answer $a_t \sim p(\cdot|f_a(s_t; \theta_a))$;
Return $T = t$ and $a_T = a_t$;

the attention-sum reader with multiple iterations by pushing the query encoding into an attention-based gate in each iteration. Iterative Alternative (IA) reader [18] produces a new query glimpse and document glimpse in each iteration and utilizes them alternatively in the next iteration. [4] further propose to extend the query-specific attention to both query-to-document attention and document-to-query attention, which is built from the intermediate results in the query-specific attention. By reading documents and enriching the query in an iterative fashion, multi-turn reasoning has demonstrated their superior performance consistently.

Our proposed approach explores the idea of using both attention-sum to aggregate candidate attention scores and multiple turns to attain a better reasoning capability. Unlike previous approaches using fixed number of hops or iterations, motivated by [14, 13], we propose a termination module in the inference. The termination module can decide whether to continue to infer the next turn after digesting intermediate information, or to terminate the whole inference process when it concludes existing information is sufficient to yield an answer. The number of turns in the inference is dynamically modeled by both a document and a query, and is generally related to the complexity of the document and the query.

3 Reasoning Networks

ReasoNets are devised to mimic the inference process of human readers. ReasoNets read a document repeatedly, with attention on different parts each time until a satisfying answer is found. As shown in Figure 1, a ReasoNet is composed of the following components:

Internal State: The internal state is denoted as S which is a vector representation of the question state. Typically, the initial state s_1 is the last-word vector representation of query by an RNN. The t-th time step of the internal state is represented by s_t. The sequence of internal state is modeled by an RNN: $s_{t+1} = \text{RNN}(s_t, x_t; \theta_s)$;

Memory: The external memory is denoted as M. It is a list of word vectors, $M = \{m_i\}_{i=1..D}$, where m_i is a fixed dimensional vector. In machine comprehensive tasks, m_i is the vector representation of each word in the doc by a bidirectional-RNN.

Attention: Attention vector x_t is generated based on the current internal state s_t and the external memory M: $x_t = f_{att}(s_t, M; \theta_x)$;

Termination Gate: Termination gate generates a stochastic random variable according to the current internal state; $t_t \sim p(\cdot|f_{tg}(s_t; \theta_{tg})))$. t_t is a binary random variable. If t_t is true, the ReasoNet stops, and the answer module executes at time step t; otherwise the ReasoNet generates an attention vector x_{t+1}, and feed into the state network to update the next internal state s_{t+1}.

Answer: The action of answer module is triggered when the termination gate variable is true: $a_t \sim p(\cdot|f_a(s_t; \theta_a))$.

In Algorithm 1, we describe the stochastic inference process of a ReasoNet. The process can be considered as a Partially Observable Markov Decision Process (POMDP) [9] in the reinforcement learning (RL) literature. The state sequence $s_{1:T}$ is hidden and dynamic, controlled by an RNN sequence model. The ReasoNet performs an answer action a_T at the T-th step, which implies that the termination gate variables $t_{1:T} = (t_1 = 0, t_2 = 0, ..., t_{T-1} = 0, t_T = 1)$. The ReasoNet learns a stochastic policy $\pi((t_t, a_t)|s_t; \theta)$ with parameters θ to get a distribution over termination actions, to

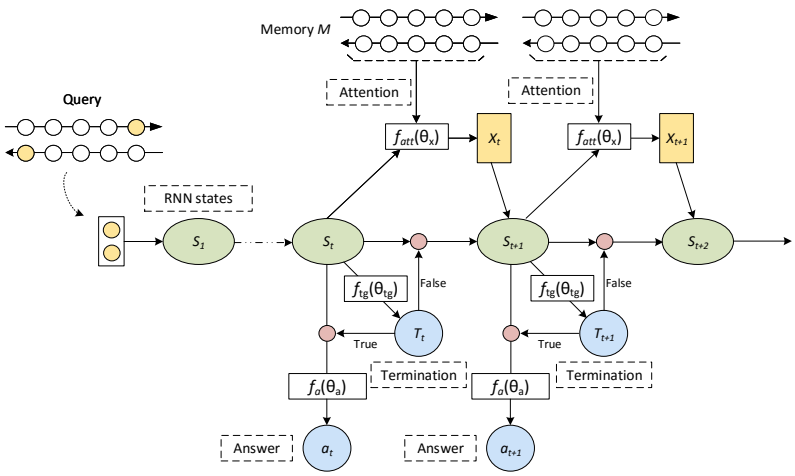


Figure 1: A ReasoNet Architecture.

continue reading or to stop, and over answer actions if the model decides to stop at the current step. The termination step T varies from instance to instance.

The parameters θ of the ReasoNet are given by the parameters of the embedding matrices W, attention network θ_x, the state RNN network θ_s, the answer action network θ_a, and the termination gate network θ_{tg}. The parameters $\theta = \{W, \theta_x, \theta_s, \theta_a, \theta_{tg}\}$ are trained by maximizing the total expect reward. The expected reward for an instance is defined as:

$$J(\theta) = \mathbb{E}_{\pi(t_{1:T}, a_T; \theta)} \left[\sum_{t=1}^{T} r_t \right]$$

The reward can only be received at the final termination step when an answer action a_T is performed. We define $r_T = 1$ if $t_T = 1$ and the answer is correct, and $r_T = 0$ otherwise. The rewards on intermediate steps are zeros, $\{r_t = 0\}_{t=1...T-1}$. J can be maximized by directly applying gradient based optimization methods. The gradient of J is given by:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi(t_{1:T}, a_T; \theta)} \left[\nabla_\theta \text{log} \pi(t_{1:T}, a_T; \theta) r_T \right]$$

We apply the REINFORCE algorithm [24] to compute $\nabla_\theta J(\theta)$:

$$\mathbb{E}_{\pi(t_{1:T}, a_T; \theta)} \left[\nabla_\theta \text{log} \pi(t_{1:T}, a_T; \theta) r_T \right] = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta) \left[\nabla_\theta \text{log} \pi(t_{1:T}, a_T; \theta)(r_T - b_T) \right]$$

where $\mathbb{A}^\dagger$ is all the possible episodes, $T, t_{1:T}, a_T$ and r_T are the termination step, termination action, answer action, and reward, respectively, for the ($t_{1:T}$, a_T) episode. b_T is called the reward baseline in the RL literature to lower variance [21]. It is common to select $b_T = \mathbb{E}_\pi [r_T]$ [20], and can be updated via an online moving average approach : $b_T = \lambda b_T + (1 - \lambda) r_T$.

However, we empirically find that above approach leads to slow convergence in training ReasoNets. Intuitively, the average baselines $\{b_T; T = 1..T_{\text{max}}\}$ are global variables independent of instances. It is hard for these baselines to capture the dynamic termination behavior of ReasoNets. In other words, ReasoNets may stop at different time steps for different instances. The adoption of a global variable without considering the dynamic variance in each instance is inappropriate. To resolve this weakness in traditional methods and account for the dynamic characteristic of ReasoNets, we propose an instance-based baseline method called "Contrastive Reward" (CR) to calculate $\nabla_\theta J(\theta)$. The basic idea of CR is to utilize an instance-based baseline assignment. We will elaborate its implementation details in Section 3.1. Empirical results show that the proposed reward schema has produced better results compared to the baseline approach.

3.1 Training Details

In the machine reading comprehension tasks, a training dataset can be simplified as a collection of triplets of query $\mathbf{q}$, passage $\mathbf{p}$, and answer $\mathbf{a}$. Say $\langle q_n, p_n, a_n \rangle$ is the n-th training instance.

The first step is to extract memory M from p_n by mapping each symbolic in the passage to a contextual representation given by the concatenation of forward and backward RNN hidden states, i.e., $m_k = [\overrightarrow{p_n}^k, \overleftarrow{p_n}^{|p_n|-k+1}]$, and extract initial state s_1 from q_n by assigning $s_1 = [\overrightarrow{q_n}^{|q_n|}, \overleftarrow{q_n}^1]$. Given M and s_1 for the n-th training instance, a ReasoNet executes $|\mathbb{A}^\dagger|$ episodes, where all possible episodes $\mathbb{A}^\dagger$ can be enumerated by setting a maximum step. Each episode generates actions and a reward from the last step: $\langle (t_{1:T}, a_T), r_T \rangle_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger}$.

Therefore, the gradient of J can be rewritten as:

$$\nabla_\theta J(\theta) = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta) \left[\nabla_\theta \log \pi(t_{1:T}, a_T; \theta)(r_T - b) \right]$$

where the baseline $b = \sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta) r_T$ is the average reward on the $|\mathbb{A}^\dagger|$ episodes for the n-th training instance. It allows different baselines for different training instances. This can be beneficial since the complexity of training instances varies significantly. Since the sum of the proposed rewards over $|\mathbb{A}^\dagger|$ episodes is zero, $\sum_{(t_{1:T}, a_T) \in \mathbb{A}^\dagger} \pi(t_{1:T}, a_T; \theta)(r_T - b) = 0$, we call it Contrastive Reward in this work. In experiments, we empirically find using $(\frac{r_T}{b} - 1)$ in replace of $(r_T - b)$ can lead to a faster convergence. Therefore, we adopt this approach to train ReasoNets in the experiments.

4 Experiments

4.1 CNN and Daily Mail Datasets

We evaluate the performance of ReasoNets on CNN and Daily Mail datasets.[2] The detailed settings of the ReasoNet model are as follows.

Vocab Size: For training our ReasoNet, we keep the most frequent $|V| = 101k$ words (not including 584 entities and 1 placeholder marker) in the CNN dataset, and $|V| = 151k$ words (not including 530 entities and 1 placeholder marker) in the Daily Mail dataset.

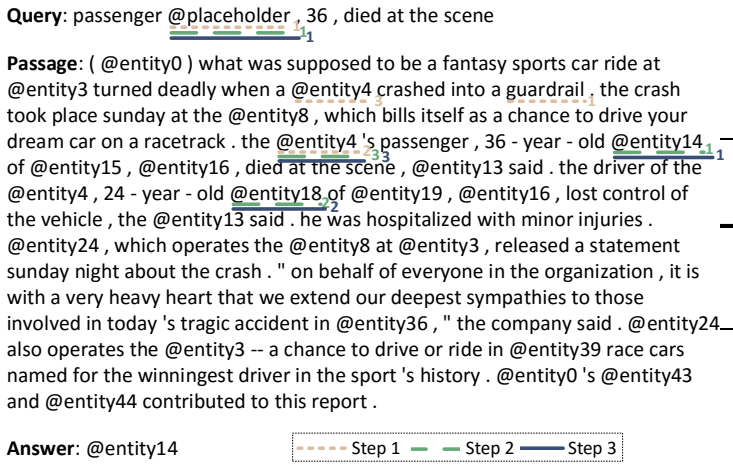
Embedding Layer: We choose word embedding size $d = 300$, and use the 300 dimensional pretrained Glove word embeddings [15] for initialization. We also apply dropout with probability 0.2 to the embedding layer.

Bi-GRU Encoder: We apply bi-directional GRU for encoding query and passage into vector representations. We set the number of hidden units to be 256 and 384 for the CNN and Daily Mail datasets, respectively. The recurrent weights of GRUs are initialized with random orthogonal matrices. The other weights in GRU cell are initialized from a uniform distribution between -0.01 and 0.01. We use a shared GRU model for both query and passage.

Memory and Attention: The memory of the ReasoNet on CNN and Daily Mail dataset is composed of query memory and passage memory. $M = (M^{query}, M^{doc})$, where M^{query} and M^{doc} are extracted from query bidirectional-GRU encoder and passage bidirectional-GRU encoder respectively. We choose projected cosine similarity function as the attention module. The attention score $a^{doc}_{t,i}$ on memory m^{doc}_i given the state s_t is computed as follows: $a^{doc}_{t,i} = \text{softmax}_{i=1,\ldots,|M^{doc}|} \gamma \cos(W^{doc}_1 m^{doc}_i, W^{doc}_2 s_t)$, where γ is set to 10. W^{doc}_1 and W^{doc}_2 are weight vectors associated with m^{doc}_i and s_t, respectively, and are joint trained in the ReasoNet. Thus, attention vector on passage is given by $x^{doc}_t = \sum_i^{|M|} a_{t,i} m^{doc}_i$. The final attention vector is the concatenation of the query attention vector and the passage attention vector $x_t = (x^{query}_t, x^{doc}_t)$. The attention module is parameterized by $\theta_x = (W^{query}_1, W^{query}_2, W^{doc}_1, W^{doc}_2)$;

Internal State Controller: We choose GRU model as the internal state controller. The number of hidden units in the GRU state controller is 256 for CNN and 384 for Daily Mail. The initial state

[2]The CNN and Daily Mail datasets are available at https://github.com/deepmind/rc-data



Step	Termination Probability	Attention Sum
1	0.0011	0.4916
2	0.5747	0.5486
3	0.9178	0.5577

Figure 2: Results of a test example 69e1f777e41bf67d5a22b7c69ae76f0ae873cf43.story from the CNN dataset. The numbers next to the underline bars indicate the rank of the attention scores. The corresponding termination probability and the sum of attention scores for the answer entity are shown in the table on the right.

of the GRU controller is set to be the last-word of the query representation by a bidirectional-GRU encoder.

Termination Module: We adopt a logistical regression to model the termination variable at each time step : $f_{tg}(s_t; \theta_{tg}) = \text{sigmoid}(W_{tg} s_t + b_{tg}); \theta_{tg} = (W_{tg}, b_{tg})$

Answer Module: We apply a linear projection from GRU outputs and make predictions on the entity candidates. Following the settings in AS Reader [8], we sum up scores from the same candidate and make a prediction. Thus, AS Reader can be viewed as a special case of ReasoNets with $T_{\max} = 1$.

Other Details: The maximum reasoning step, $T_{\max}$ is set to 5 in experiments on both CNN and Daily Mail datasets. We use ADAM optimizer [10] for parameter optimization with an initial learning rate of 0.0005, $\beta_1 = 0.9$ and $\beta_2 = 0.999$; The absolute value of gradient on each parameter is clipped within 0.001. The batch size is 64 for both CNN and Daily Mail datasets. For each batch of the CNN and Daily Mail datasets we randomly reshuffle the assignment of named entities [6]. This forces the model to treat the named entities as semantically meaningless labels. In the prediction of test cases, we randomly reshuffle named entities up to 4 times, and report the averaged answer. Models are trained on GTX TitanX 12GB. It takes 7 hours per epoch to train on the Daily Mail dataset and 3 hours per epoch to train on the CNN dataset. The models are usually converged within 6 epochs on both CNN and Daily Mail datasets.

Table 1 shows the performance of all the existing single model baselines and our proposed ReasoNet. By capturing multi-turn reasoning and learning to stop reading a paragraph, we have achieved the state-of-the-art results in both CNN and Daily Mail datasets. To further understand the inference process of the ReasoNet, Figure 2 shows a test example of the CNN dataset. The model initially focuses on wrong entities with low termination probability. In the second and third steps, the model focuses on the right clue with higher termination probability. Interestingly, we also find that query attention focuses on the placeholder token throughout all the steps.

4.2 Graph Reachability Task

Recent analysis and results [3] on the cloze-style machine comprehension tasks have suggested some simple models without multi-turn reasoning can achieve reasonable performance. Based on these results, we construct a synthetic structured Graph Reachability dataset[3] to evaluate longer range machine inference and reasoning capability, since we expect ReasoNets have the capability to handle long range relationships.

We generate two synthetic datasets: a small graph dataset and a large graph dataset. In the small graph dataset, it contains $500K$ small graphs, where each graph contains 9 nodes, and 16 direct edges

[3]The dataset is available at https://github.com/MSRDL/graph_reachability_dataset

Table 1: The performance of Reasoning Network on CNN and Daily Mail dataset.

	CNN		Daily Mail	
	valid	test	valid	test
Deep LSTM Reader [6]	55.0	57.0	63.3	62.2
Attentive Reader [6]	61.6	63.0	70.5	69.0
MemNets [7]	63.4	66.8	-	-
AS Reader [8]	68.6	69.5	75.0	73.9
Stanford AR [3]	72.2	72.4	76.9	75.8
DER Network [11]	71.3	72.9	-	-
Iterative Attention Reader [18]	72.6	73.3	-	-
EpiReader [22]	**73.4**	74.0	-	-
GA Reader [5]	73.0	73.8	76.7	75.7
AoA Reader [4]	73.1	74.4	-	-
ReasoNet	72.9	**74.7**	**77.6**	**76.6**

Table 2: Reachability statistics of the Graph Reachability dataset.

	Small Graph				Large Graph			
Reachable Step	No Reach	1–3	4–6	7–9	No Reach	1–3	4–6	7–13
Train (%)	44.16	42.06	13.51	0.27	49.02	25.57	21.92	3.49
Test (%)	45.00	41.35	13.44	0.21	49.27	25.46	21.74	3.53

to randomly connect pairs of nodes. The large graph dataset contains $500K$ graphs, where each graph contains 18 nodes, and 32 random direct edges. Duplicated edges are removed. Table 2 shows the graph reachability statistics on the two datasets.

In Table 3, we show examples of a small graph and a large graph in the synthetic dataset. Both graph and query are represented by a sequence of symbols. In the experiment, we use a 100-dimensional embedding vector for each symbol, and bidirectional-LSTM with 128 and 256 cells for query and graph embedding in the small and the large graph datasets, respectively. The last states of bidirectional-LSTM on query are concatenated to be the initial internal state $s_1 = [\overrightarrow{q}^{|q|}, \overleftarrow{q}^{1}]$ in the ReasoNet. Another bidirectional-LSTM on graph description maps each symbol g^i to a contextual representation given by the concatenation of forward and backward LSTM hidden states $m_i = [\overrightarrow{g}^{i}, \overleftarrow{g}^{|g|-i+1}]$. The final answer is either "Yes" or "No" and hence logistical regression is used as the answer module: $a_t = \sigma(W_a s_t + b_a)$; $\theta_a = (W_a, b_a)$. We apply another logistical regression as the termination gate module: $t_t = \sigma(W_{tg} s_t + b_{tg})$. The maximum reasoning step $T_{\max}$ is set to 15 and 25 for the small graph and large graph dataset, respectively.

We study the effectiveness of the termination gate in ReasoNets. We denote "`ReasoNet`" as a standard ReasoNet with termination gate, as described in Section 3.1. If we remove the termination gate, and just simply use the last state answer action as the final answer, say $\hat{a} = a_{T_{\max}}$ ($T_{\max}$ is the maximum reasoning step), denoted as "`ReasoNet-Last`". To study the effectiveness of multi-turn reasoning, we choose "`ReasoNet-`$T_{\max} = 2$", which only has single-turn reasoning, as a baseline.

In Table 4, we report the performance of `ReasoNet`, `ReasoNet-Last` and `ReasoNet-`$T_{\max} = 2$ models on the Graph Reachability dataset. The `ReasoNet-Last` model performs well on the small graph dataset, and it obtains 100% accuracy. However, the `ReasoNet-Last` model fails to learn on the large graph dataset, as the task becomes much more challenging. Meanwhile, the `ReasoNet` model converges faster than the `ReasoNet-Last` model. The `ReasoNet` model converges in 20 epochs in the small graph dataset, and 40 epochs in the large graph dataset, while the `ReasoNet-Last` model converges around 40 epochs in the small graph dataset, and 70 epochs in the large graph dataset. The results suggest that the termination gate variable in the ReasoNet is helpful when training with sophisticated examples, and makes models converge faster. Both the `ReasoNet` and `ReasoNet-Last` models perform better than the `ReasoNet-`$T_{\max} = 2$ model, which demonstrates the importance of multi-turn reasoning. To further understand the inference process in ReasoNets, we present two examples of the graph reachability results in appendix A.

Table 3: Small and large random graph in the Graph Reachability dataset. Note that "A –> B" represents an edge connected from A to B and the # symbol is used as a delimiter between different edges.

	Small Graph	Large Graph
Graph Description	0 –> 0 # 0 –> 2 # 1 –> 2 # 2 –> 1 # 3 –> 2 # 3 –> 3 # 3 –> 6 # 3 –> 7 # 4 –> 0 # 4 –> 1 # 4 –> 4 # 5 –> 7 # 6 –> 0 # 6 –> 1 # 7 –> 0 #	0 –> 17 # 1 –> 3 # 1 –> 14 # 1 –> 6 # 2 –> 11 # 2 –> 13 # 2 –> 15 # 3 –> 7# 5 –> 0 # 5 –> 7 # 6 –> 10 # 6 –> 5# 7 –> 15 # 7 –> 7 # 8 –> 11 # 8 –> 7 # 10 –> 9 # 10 –> 6 # 10 –> 7 # 12 –> 1 # 12 –> 12 # 12 –> 6 # 13 –> 11 # 14 –> 17 # 14 –> 14 # 15 –> 10 # 16 –> 2 # 17 –> 4 # 17 -> 7 #
Query	7 –> 4	10 –> 17
Answer	No	Yes

Table 4: The performance of Reasoning Network on the Graph Reachability dataset.

	Small Graph			Large Graph		
	ROC-AUC	PR-AUC	Accuracy	ROC-AUC	PR-AUC	Accuracy
`ReasoNet-`$T_{\texttt{max}} = 2$	0.9638	0.9677	0.8961	0.8477	0.8388	0.7607
`ReasoNet-Last`	**1**	**1**	**1**	0.8836	0.8742	0.7895
`ReasoNet`	**1**	**1**	**1**	**0.9988**	**0.9989**	**0.9821**

5 Conclusion

In this paper, we propose ReasoNets that dynamically decide whether to continue or to terminate the inference process in machine comprehension tasks. Using reinforcement learning with the proposed contractive reward, our proposed model achieves the start-of-the-art results in machine comprehension datasets, including unstructured CNN and Daily Mail datasets, and a proposed structured Graph Reachability dataset. For future work, ReasoNets can be generalized to other tasks that requires reasoning capability, such as question answering and knowledge graph inference.

Acknowledgments

We thank Ming-Wei Chang, Li Deng, Lihong Li, and Xiaodong Liu for their thoughtful feedback and discussions.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations*, 2015.

[2] Léon Bottou. From machine learning to machine reasoning. *Machine Learning*, 94(2):133–149, 2014.

[3] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the CNN / Daily Mail reading comprehension task. In *ACL*, 2016.

[4] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *CoRR*, abs/1607.04423, 2016.

[5] Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *CoRR*, abs/1606.01549, 2016.

[6] Karm Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

[7] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The Goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of the International Conference on Learning Representations*, 2016.

[8] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *arXiv:1603.01547v1 [cs.CL]*, 2016.

[9] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015.

[11] Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies (NAACL-HLT)*, 2016.

[12] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the International Conference on Machine Learning*, 2016.

[13] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pp. 2204–2212, 2014.

[14] Rodrigo Nogueira and Kyunghyun Cho. Webnav: A new large-scale task for natural language based sequential decision making. In *Advances in Neural Information Processing Systems*, 2016.

[15] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100, 000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[17] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, 2013.

[18] Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. Iterative alternating neural attention for machine reading. *CoRR*, abs/1606.02245, 2016.

[19] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

[20] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1999.

[21] Richard Stuart Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, 1984.

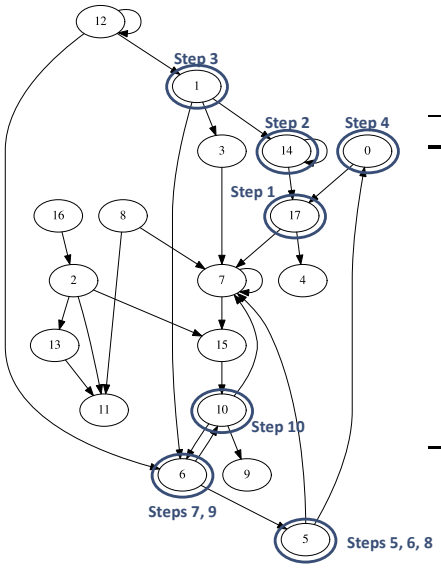
[22] Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the EpiReader. In *EMNLP*, 2016.

[23] Dirk Weissenborn. Separating answers from queries for neural reading comprehension. *CoRR*, abs/1607.03316, 2016.

[24] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.

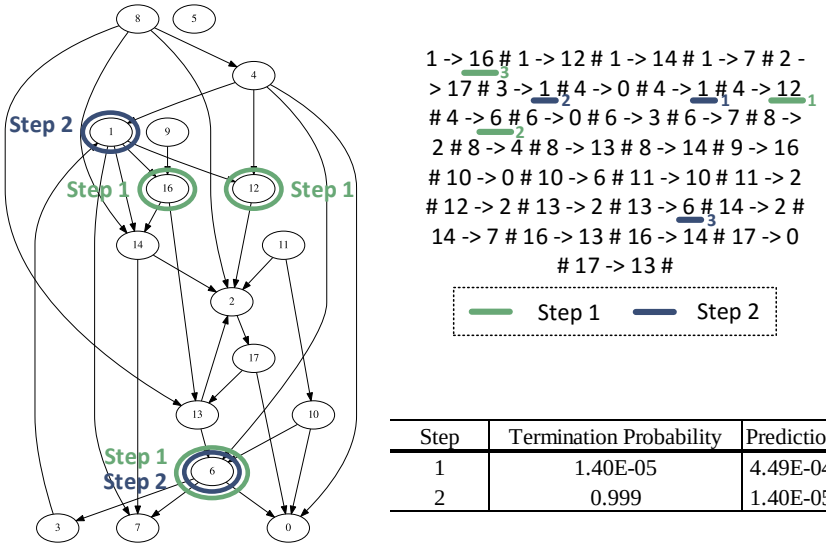
A Examples of Graph Reachability Results in ReasoNets

Figures 3 and 4 show test examples of the large graph dataset. In Figure 3, we observe that the model does not make a firm prediction till step 9. The highest attention word at each step shows the reasoning process of the model. Interestingly, the model starts from the end node (17), traverses backward till finding the starting node (10) in step 9, and makes a firm termination prediction. On the other hand, in Figure 4, the model learns to stop in step 2. In step 1, the model looks for neighbor nodes (12, 6, 16) to 4 and 9. Then, the model gives up in step 2 and predict "No". All of these demonstrate the dynamic termination characteristic and potential reasoning capability of ReasoNets.



Step	Termination Probability	Prediction
1	1.00E-06	0.172
2	1.00E-06	0.625
3	1.00E-06	0.752
4	1.00E-06	0.202
5	1.00E-06	0.065
6	1.00E-06	0.041
7	2.30E-06	0.137
8	0.0017	0.136
9	0.49	0.761
10	0.99	0.927

Figure 3: An example of graph reachability result, given a query "10 –> 17" (Answer: Yes). The red circles highlight the nodes/edges which have the highest attention in each step. The corresponding termination probability and prediction results are shown in the table. The model terminates at step 10.



Step	Termination Probability	Prediction
1	1.40E-05	4.49E-04
2	0.999	1.40E-05

Figure 4: An example of graph reachability result, given a query "4 –> 9" (Answer: No). The numbers next to the underline bars indicate the rank of the attention scores. The corresponding termination probability and prediction results are shown in the table.

Pre-Wiring and *Pre-Training*: What does a neural network need to learn truly general identity rules?

Raquel G. Alhama and **Willem Zuidema**
Institute for Logic, Language and Computation
University of Amsterdam, The Netherlands
{rgalhama,w.h.zuidema}@uva.nl

Abstract

In an influential paper, Marcus et al. [1999] claimed that connectionist models cannot account for human success at learning tasks that involved generalization of abstract knowledge such as grammatical rules. This claim triggered a heated debate, centered mostly around variants of the Simple Recurrent Network model [Elman, 1990]. In our work, we revisit this unresolved debate and analyze the underlying issues from a different perspective. We argue that, in order to simulate human-like learning of grammatical rules, a neural network model should not be used as a *tabula rasa*, but rather, the initial wiring of the neural connections and the experience acquired prior to the actual task should be incorporated into the model. We present two methods that aim to provide such initial state: a manipulation of the initial connections of the network in a cognitively plausible manner (concretely, by implementing a "delay-line" memory), and a pre-training algorithm that incrementally challenges the network with novel stimuli. We implement such techniques in an Echo State Network [Jaeger, 2001], and we show that only when combining both techniques the ESN is able to learn truly general identity rules.

1 Introduction

One of the crucial aspects of language is that it allows humans to produce and understand an unlimited number of utterances. This is possible because language is a rule-governed system; for instance, if we know that the English present participle is formed by appending *-ing*, then we readily *generalize* this pattern to novel verbs. Accounting for how humans learn these abstract patterns, represent them and apply them to novel instances is the central challenge for cognitive science and linguistics. In natural languages there is an abundance of such phenomena, and as a result linguistics has been one of the main battlegrounds for debates between proponents of symbolic and connectionists accounts of cognition. One of the most heated debates was concerned with accounting for the regular and irregular forms of the English past tense. Rumelhart and McClelland [1986] proposed a connectionist model that allegedly accounted for the regular and irregular forms of the past tense. However, this model was fiercely critized by Steven Pinker and colleagues [Pinker and Prince, 1988, Pinker, 2015], who held that rules are essential to account for regular forms, while irregular forms are stored in the lexicon (the 'Words-and-Rules' theory).

A similar debate emerged with the publication of Marcus et al. [1999], this time centered on experimental results in Artificial Grammar Learning. The authors showed that 7 month old infants generalize to novel instances of simple ABA, ABB or AAB patterns after a short familiarization. Crucially, this outcome could not be reproduced by a Simple Recurrent Network (SRN) [Elman, 1990], a result that was interpreted by the authors as evidence in favour of a symbol-manipulating system:

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

> Such networks can simulate knowledge of grammatical rules only by being trained on all items to which they apply; consequently, such mechanisms cannot account for how humans generalize rules to new items that do not overlap with the items that appear in training. [Marcus et al., 1999, p. 79]

This claim triggered many replies, some of which proposed variations of the original model. However, in this debate the issues of whether neural networks are capable at all of *representing* general rules, of whether backpropagation is capable of *finding* these general rules from an arbitrary initial state or only from an appropriately chosen initial state are sometimes conflated. The latter issue – what initial state does a neural network model need to have success in the experiment – has, in our view, not received enough attention (but see Seidenberg and Elman [1999a], Altmann [2002]). This will be therefore the focus of this paper, in which we explore two directions. First, we ask which initial values of the connection weights could encourage generalization while remaining cognitively plausible (*pre-wiring*); second, we investigate the role of previous experience in creating an initial state in the network that would facilitate generalization (*pre-training*). We employ a prewiring and a pretraining technique in an Echo State Network (ESN) (Jaeger, 2001), and show that only when combining both techniques the ESN is able to accurately generalize to novel items.

2 Background

2.1 Empirical Data

Marcus et al. [1999] investigate the generalization abilities of 7 month old infants by conducting three Artificial Grammar Learning experiments. In their first experiment, the participants are familiarized to syllable triplets that follow a certain grammar: ABA for a randomly assigned group of infants, and ABB for the other. The stimuli contain 16 different triplets, each repeated 3 times. Those triplets are arranged in a 2-min. auditory speech stream, such that syllables are separated by a pause of 250 ms, and triplets of syllables are separated by 1s.

After the familiarization, the infants participate in a test phase, in which their looking times (to the speaker device that plays the stimuli) are recorded. The speaker plays a randomized set of triplets from both grammars, in order to see if infants can discriminate between them. Crucially, the test triplets contain syllables that were not used in the familiarization stimuli.

The results show a statistically significant difference between mean looking times to consistent and inconsistent grammars in both group of infants. The authors then conclude that infants can discriminate among ABA and ABB grammars.

Table 1: Stimuli used in experiment 2 in Marcus et al. [1999].

	Familiarization				Test
ABA	le di le le je le le li le le we le	wi di wi wi je wi wi li wi wi we wi	ji di ji ji je ji ji li ji ji we ji	de di de de je de de li de de we de	ba po ba ko ga ko
ABB	le di di le je je le li li le we we	wi di di wi je je wi li li wi we we	ji di di ji je je ji li li ji we we	de di di de je je de li li de we we	ba po po ko ga ga
3x triplet (random order)					

This experiment was repeated with a more carefully controlled set of syllables, which we report in table 1. Infants also exhibit significantly different behavioural responses in this second experiment. Finally, an additional experiment was performed, in this case using AAB vs. ABB grammars, in order to determine whether the rule learnt before was simply the presence or absence of an immediate repetition. Infants also showed significantly different responses in this experiment.

In the light of these results, the authors concluded that: (i) 7 m.o. infants can extract grammar-like rules, (ii) they can do it not based solely on statistical information (as would be evidenced from the additional controls in experiment 2, and (iii) the extracted rule is not merely the presence or absence of an immediate repetition.

2.2 Generalization and Neural Networks

Marcus [1998] argues that certain types of generalizations are unattainable for certain types of neural networks: concretely, those that lie *outside the training space*. The author defines *training space* as the combination of all feature values that network has witnessed during training. If there exist feature values that have never appeared during training, any item displaying that feature value lies outside the training space. For neural networks that are trained with the backpropagation algorithm, generalization to items outside the training space is, according to the author, extremely unlikely to occur due to what he calls *training independence*, which stands for the fact that the algorithm updates the weights of nodes independently of the activations of other nodes in the same layer.

In Marcus et al. [1999], the authors provide empirical evidence in support of this idea, by simulating the presented experiment in a Simple Recurrent Network (SRN) [Elman, 1990], a neural network architecture that incorporates an additional context layer that maintains an exact copy of the hidden layer and presents it to the network in the subsequent timestep, providing the model with memory in this way. The SRN is trained to predict the next syllable in the familiarization stimuli, and then tested on its ability to predict the final syllable of test items consistent with the familiarization grammar. This model failed to produce the correct predictions, confirming the hypothesis of the researchers.

Some following publications proposed to change the encoding of the input (Christiansen and Curtin [1999], Christiansen et al. [2000], Eimas [1999], Dienes et al. [1999], Altmann and Dienes [1999], [McClelland and Plaut, 1999]), the task (Seidenberg and Elman [1999a], Seidenberg and Elman [1999b], the neural network architecture (Shultz [1999], Sirois et al. [2000], Shultz and Bale [2001]), or – relevant to our work — incorporating some form of pre-training (Seidenberg and Elman [1999a], Altmann [2002]). Many of these models were subject of criticism by Marcus (Marcus [1999a],Marcus [1999b],Marcus [1999c],Marcus [1999d]), who argued that the models either involved some form of symbolic manipulation or did not adequately represent the experiment. About the model of Altmann [2002], which involves pre-training similar to the regime we explore in section 5, Marcus [1999e] points out, without giving any details, that, even if the model distinguishes grammatical from ungrammatical stimuli to some degree, it is unclear whether the model can actually learn the underlying general rule or discovers some heuristic that weakly correlates with it. In our work, we employ a neural network architecture that was not previously explored for this task (an Echo State Network, a type of Reservoir Computing network), and we report additional performance measures that tell us more about how general the learned rules are.

3 Simulations with a Simple Recurrent Network

Before presenting our simulations with the ESN model, we report our replication of the original simulations. We implement a Simple Recurrent Network as described in Elman [1990], and we train it to predict the next syllable in the input. As in Marcus et al., we use distributional encoding of phonetic features (based on Plunkett and Marchman [1993]). But unlike the original simulations, we do not encode the pause between triplet as an additional symbol; instead, we do not update the weights in the network after the last syllable of a triplet is presented.

In order to remain close to the test used in the experiments with infants, we test the network on both consistent and inconsistent sequences. We take the predicted vector for the third syllable of each triplet, and we find the closest vector that corresponds to one of the seen syllables (both from training and from test). We then evaluate whether the accuracy for consistent and inconsistent triplets is significantly different (for 16 runs of the model, equivalent to the number of infants in the experiment).

The test set used in the original experiments, as can be seen in Table 1, is based solely in two triplet types of each grammar. For this reason, we also evaluate our model with an extended test set that contains 5 additional random novel syllables of each type (A and B), consisting therefore of 25 test triplets.

We try 160 parameter settings for each familiarization grammar, varying the hyperparameters of the model: the size of the hidden layer, the learning rate and the number of epochs [1] . Figure 1 shows the proportion of these runs that yield a significant difference in the predictions for the two classes of test items (those that are consistent with the grammar used in training and those which are not). For the

[1]We found that the values of the three hyperparameters had a significant effect on the accuracy of the predicted syllables in the test.

responses that are significantly different, we separate those for which the neural network responds better to the consistent grammar (in white) from those in which the inconsistent grammar is favoured (in grey).

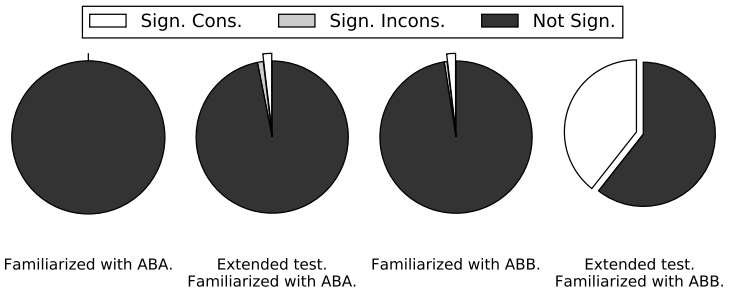


Figure 1: Proportion of parameter settings that yield significant (white), non-significant (dark grey) and inconsistently significant (light grey) responses, for simulations with an SRN.

As shown in the graphic, most of the simulations yield non-significant responses between grammars, in spite of a notable proportion of significant responses in the ABB condition for the extended test, possibly due to the fact that immediate repetitions are easier to learn[2]. We therefore confirm that the Simple Recurrent Network does not reproduce the empirical findings.

4 Simulations with an Echo State Network

Recurrent Neural Networks, such as the SRN, can be seen as implementing memory: through gradual changes in synaptic connections, the network learns to exploit temporal regularities for the function it is trained on. An alternative way to learn time-dependent relations is that offered by Reservoir Computing (RC) approaches, such as the Liquid State Machine [Maass et al., 2002] and the model adopted here, the Echo State Network (ESN) [Jaeger, 2001, Frank and Čerňanskỳ, 2008]. In RC models, the weights in the hidden layer (which is dubbed "reservoir") remain untrained, but – if satisfying certain constraints (the so-called "Echo State Property", which depends on the scaling of the weights in the reservoir based on the spectral radius parameter) – the dynamics exhibited by the reservoir "echo" the input sequence: some memory of the input lingers on for some time in the recurrent connections. In other words, the state of the reservoir depends on the fading history of the input; after a long enough input, the initial state does not determine the final states of the network.

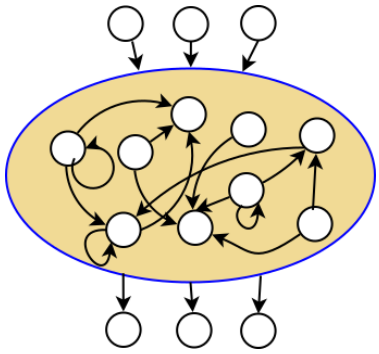


Figure 2: The Echo State Network.

The formalization of the ESN model is as follows. For an input u at time t, the activation x of the nodes in the reservoir is defined as:

$$x(t) = f(W^{in} \cdot u(t) + W^{res} \cdot x(t-1)) \qquad (1)$$

where W^{in} are the input weights, W^{res} are the internal weights of the reservoir, and f is a non-linear function, generally $tanh$.

The activation of the output is defined as:

$$y(t) = f^{out}(W^{out} \cdot x(t)) \qquad (2)$$

where W^{out} are the weights that connect the reservoir with the output nodes, and f^{out} is a function, which might be different from the function applied to the reservoir; in fact, it often consists on a simple identity function.

We implement a basic ESN with tanh binary neurons, and we follow the same procedure described in section 3 to train the network with backpropagation[3]. We try 200 parameter settings for each

[2]This was also observed in the SRN model in Altmann [2002].

[3]We have also run simulations with Ridge Regression, with similar results.

familiarization grammar, varying the hyperparameters of the model: the number of nodes in the reservoir, the input scaling, the spectral radius, learning rate and epochs.[4] Figure 3 shows the proportion of these runs that yield a significant difference in the predictions.

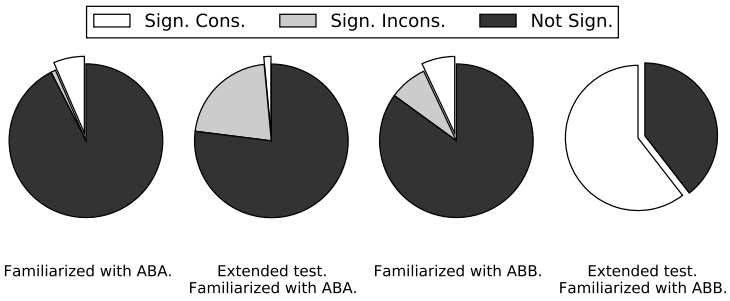


Figure 3: Proportion of parameter settings that yield significant (white), non-significant (dark grey) and inconsistently significant (light grey) responses, for the simulation with the basic ESN.

As can be seen, the results based on the Marcus et al. test set differ greatly from those in our extended test. This confirms our intuition that the amount of test items is crucial for the evaluation. For this reason, we base our analysis of the behaviour of the model in the extended test; however, it is important to notice that the amount of test items could have also played a role in the actual experiments with infants (see also section 7).

The plots of the extended test condition clearly show an assymetry between the grammars: more than half of the parameter settings yield significant responses for the ABB, while in the case of ABA, less than a quarter of the simulations are significant, and most of them are actually favouring the inconsistent grammar, which is precisely ABB. As mentioned before, the reason for this assymetry is probably due to the fact that immediate repetitions are easier to learn, since they are less affected from the decay of the activation function; for now, it suffices to say that the behaviour of the model towards ABA does not suggest that it could be a potential explanation for the experimental results.

5 *Pre-Wiring*: Delay Line Memory

In order to succeed in the prediction task, the model must predict a syllable that is identical to one presented before. In the previous simulations, we relied on the memory that ESNs offer through the recurrent connections and the Echo Property [Jaeger, 2002]. However, there exist several computational alternatives that brains may use to implement memory [Chaudhuri and Fiete, 2016]. We now explore one such model of memory: a delay line.

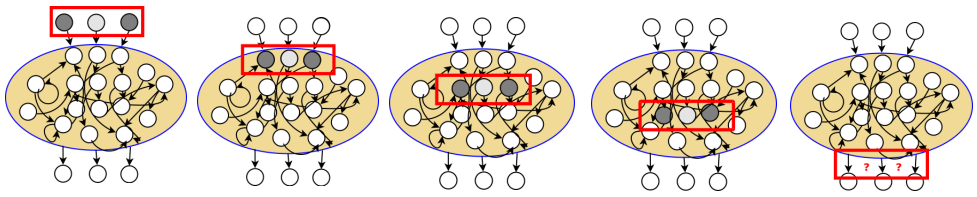


Figure 4: Depiction of five timesteps in the DeLi-ESN. The highlighted nodes show the activations in the delay line (the activation of the rest of the nodes is not illustrated).

Computationally, a delay line is a mechanism that ensures the preservation of the input by propagating it in a path ("line") that implements a delay. In the brain, delay lines have been proposed as part of the sound localization system [Jeffress, 1948], and they have been identified through intracellular

[4] We found that the values of the input scaling and the learning rate had a significant effect on the accuracy of the predicted syllables in the test.

recordings in the barn owl brain [Carr and Konishi, 1988]. In a neural network, a delay line is naturally implemented by organizing a subnetwork of neurons in layers, with "copy connections" (with weights 1 between corresponding nodes and 0 everywhere else) connecting each layer to the next. In this way, the information is kept in the network for as many timesteps as dedicated layers in the network (see figure 4).

We implement a delay line memory in the ESN (creating thus a new model that we call *DeLi-ESN*) by imposing this layer structure in the reservoir. We run 1200 combinations of parameter settings with the DeLi-ESN, including also a parameter that establishes some amount of noise to add to the weights of the reservoir. In this way, some models contain a less strict delay line; the greater the noise, the closer the model is to the original ESN.

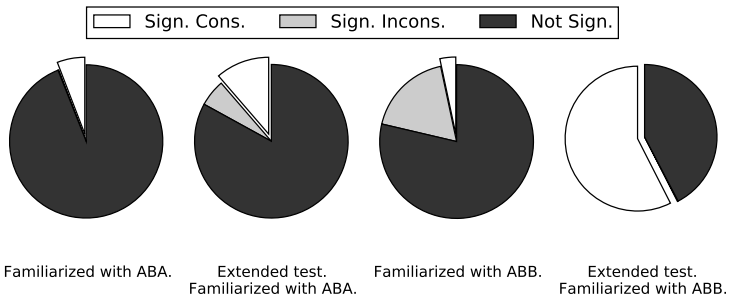


Figure 5: Proportion of parameter settings that yield significant, non-significant and inconsistently significant responses in the tests, for the simulation with DeLi-ESN.

The results, illustrated in Figure 5, show an increased number of significant responses (in favour of the consistent grammar) for the extended test of ABA familiarization: the addition of the delay line memory indeed helps in the detection of the identity relation. But in spite of the positive effect of the delay line, we need to ask ourselves to what extent these results are satisfactory. The pie plots show the likelihood of obtaining the results that Marcus et al. found in their experiments with our model, and in order to do so, we use the same measure of success (i.e. whether the responses for each grammar are significantly different). However, the models hardly ever produce the correct prediction[5]. For this reason, in the next section, we adopt a stricter measure of success. We discuss this issue further in section 7.

6 *Pre-Training*: Incremental-Novelty Exposure

The infants that participated in the original experiment had surely been exposed to human speech before the actual experiment; however, in most computational simulations this fact is obviated. We hypothesize that prior perceptual experience could have triggered a bias for learning abstract solutions: since the environment is variable, infants may have adapted their induction mechanism to account for novel instances. We now propose a method to pre-train a neural network that aims to incorporate this intuition.

In this training regime —which we call Incremental-Novelty Exposure, or INE for short— we iteratively train and test our model; so for a certain number of iterations i, the model is trained and tested i times, with the parameters learnt in one iteration being the initial state of the next iteration. The test remains constant in each of these iterations; however, the training data is slightly modified from one iteration to the next. The first training set is designed according to the Marcus et al. stimuli: 4 syllables of type A and 4 syllables of type B are combined according to the pattern of the familiarization grammar (ABA or ABB). In the second iteration, one syllable of type A and one of type B are deleted (that is, all the triplets involving those syllables are removed from the training set), and a new syllable of type A and one of type B are incorporated, such that new

[5]We find that the values of the input scaling, learning rate, spectral radius, reservoir size, and reservoir noise each have a significant effect on the accuracy of the predicted syllables in the test, although exact prediction accuracy remains low (rarely above 12% for ABA and above 20% for ABB familiarization) even for the best combination of parameters.

triplets of the familiarization pattern are generated with the new syllables (combined as well with the already-present syllables). Therefore, for each training iteration, the model is exposed to a similar training set as the previous iteration, but there is a small amount of novelty.

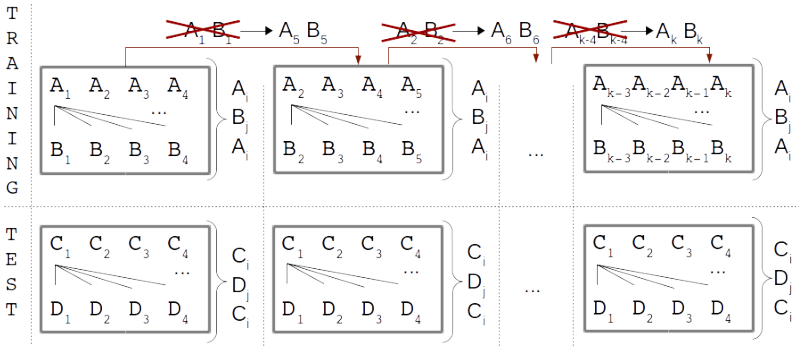


Figure 6: Depiction of the Incremental-Novelty Exposure. Barred syllables are removed from the training set after the training of the corresponding iteration has finished (so they do not remain in the training set of the next iteration).

We simulate 600 different hyperparameter configurations, varying the reservoir size, noise in the delay line, input scaling, spectral radius, learning rate, and epochs. Figure 6 illustrates how the mean accuracy evolves at each stage of the INE procedure of one representative run. As it can be seen in the graphs, the accuracy is really low in the beginning (corresponding to a simulation without pre-training) but, with more iterations –and thus with more novel items incorporated in the training set–, the model becomes better, presumably by finding a more general solution.

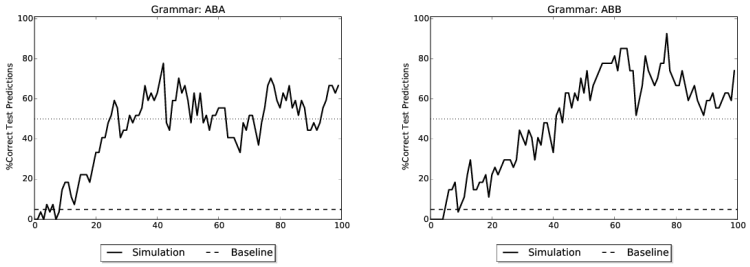


Figure 7: Performance over time 100 epochs with incremental novelty exposure, for one representative run in the ABA familiarization condition (left) and one in the ABB condition (right).

In order to test that these results are robust, we compute the mean over the accuracy for the last quarter of the tests (in our case, the last 25 tests, corresponding to the rightmost curve in the graph), for a few runs. The results are fairly similar in each run, as can be seen in figure 8.

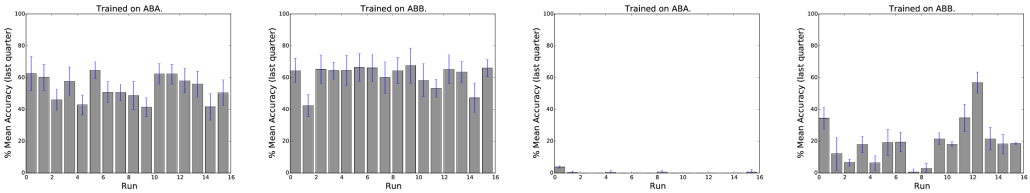


Figure 8: Mean accuracy for the tests in the last quarter simulation in the INE simulation, for DeLi-ESN (left) and basic ESN (right).

Both graphs show that the combination of the DeLi-ESN and the INE drastically boosts the generalization capabilities of the ESN. However, we should identify what is the contribution of the DeLi-ESN. Figure 8 shows the mean accuracy (again, for the last quarter of the regime), for 16 runs

of the basic ESN in the INE regime. The effect of the delay line memory is blatantly clear: when removed, the accuracy is close to 0 for ABA, and mostly around 20% for ABB.

7 Discussion

The Marcus et al. publication conveyed two bold statements: first, that infants spontaneously extract identity rules from a short familiarization, and second, that neural networks are doomed to fail in such simple task. Our work suggests that both the initial optimism for the generalization abilities of infants and the pessimism towards neural networks were overstated.

Our work investigates the initial conditions that a neural network model should implement in order to account for the experimental results. First, we have proposed that networks should be *pre-wired* to incorporate a bias towards memorization of the input, which we have implemented as a delay line. With such *pre-wiring*, the model yields a notable proportion of significantly different responses between grammars. But despite such apparent success, the accuracy of the model in syllable prediction is very low.

Therefore, even though the successful discrimination between grammars is generally understood as abstraction of the underlying rule, our results show that significantly different responses can easily be found in a model that has not perfectly learnt such a rule. The corollary is that the generalization abilities of infants may have been overestimated; as a matter of fact, null results in similar experiments also point in that direction (see for instance footnote 1 in Gerken [2006]; also Geambasu&Levelt, p.c.).

But can neural networks go beyond grammar discrimination and accurately predict the next syllable according to a generalized rule? Our work shows that this can be achieved when prior experience is incorporated in the model. We have hypothesized that, from all the information available in the environment, it is the gradual exposure to novel items what enhances generalization. This particular hypothesis deviates from related studies in which (i) an SRN was pre-trained to learn the relation of *sameness* between syllables [Seidenberg and Elman, 1999a], and (ii) an SRN was pre-trained with a set of sentences generated from a uniformly sampled vocabulary. Although the data used in our pre-training is less realistic than that used by Altmann [2002], our evaluation method is more strict (since we aim to test for accuracy rather than discrimination); for this reason, we first need to evaluate a model with a more constrained input. The next step in future work should be to explore whether the same results can be obtained when input data involving gradual novelty is generated from a grammar unrelated to the actual experiment.

Finally, from the perspective of the symbol vs. associations debate, at some abstract level of description, the delay line may be interpreted as providing the model with variables (that is, the dedicated group of nodes that encode the input at a certain time may be seen as a register) and the symbolic operation of "copy". It should be noted though that these groups of nodes are not isolated, and therefore, the learning algorithm needs to discover the structure in order to make use of it. Furthermore, it is uncontroversial that items are kept in memory for a certain lapse of time, so this structure is unlikely to constitute the core of the symbolic enterprise. If nevertheless our model is seen as compatible with the theory of rules-over-variables, our approach would then constitute a fundamental advancement for the field in providing a unifying model in which both theories can see their proposals reflected.

Acknowledgments

This research was funded by a grant from the Netherlands Organisation for Scientific Research (NWO), Division of Humanities, to Levelt, ten Cate and Zuidema (360-70-450).

References

Gerry T.M. Altmann. Learning and development in neural networks–the importance of prior experience. *Cognition*, 85(2):B43–B50, 2002.

Gerry T.M. Altmann and Zoltán Dienes. Technical comment on rule learning by seven-month-old infants and neural networks. *Science*, 284(5416):875–875, 1999.

Catherine E. Carr and Masakazu Konishi. Axonal delay lines for time measurement in the owl's brainstem. *Proceedings of the National Academy of Sciences*, 85(21):8311–8315, 1988.

R. Chaudhuri and I. Fiete. Computational principles of memory. *Nature neuroscience*, 19(3):394–403, 2016.

Morten H. Christiansen and Suzanne Curtin. Transfer of learning: rule acquisition or statistical learning? *Trends in Cognitive Sciences*, 3(8):289 – 290, 1999.

Morten H. Christiansen, C.M. Conway, and Susan Curtin. A connectionist single mechanism account of rule-like behavior in infancy. In *Proceedings of the 22nd annual conference of the cognitive science society*, pages 83–88, 2000.

Zoltán Dienes, Gerry Altmann, and Shi-Ji Gao. Mapping across domains without feedback: A neural network model of transfer of implicit knowledge. *Cognitive Science*, 23(1):53–82, 1999.

P. Eimas. Do infants learn grammar with algebra or statistics? *Science*, 284(5413):435, 1999.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

Stefan L. Frank and M Čerňanskỳ. Generalization and systematicity in echo state networks. In & V.M. Sloutsky B.C. Love, K. McRae, editor, *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 733–738. Cognitive Science Society, 2008.

LouAnn Gerken. Decisions, decisions: infant language learning when multiple generalizations are possible. *Cognition*, 98(3):B67 – B74, 2006.

Herbert Jaeger. The ''echo state'' approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.

Herbert Jaeger. Short term memory in echo state networks. Technical report, German National Research Center for Information Technology, 2002.

Lloyd A. Jeffress. A place theory of sound localization. *Journal of comparative and physiological psychology*, 41(1):35, 1948.

Wolfgang Maass, Thomas Natschläger, and Henry Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

G. F. Marcus, S. Vijayan, S.B. Rao, and P.M. Vishton. Rule learning by seven-month-old infants. *Science*, 283 (5398):77–80, 1999.

Gary F. Marcus. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282, 1998.

Gary F. Marcus. Connectionism: with or without rules?: Response to J.L. McClelland and D.C. Plaut (1999). *Trends in Cognitive Sciences*, 3(5):168 – 170, 1999a.

Gary F. Marcus. Do infants learn grammar with algebra or statistics? Response to Seidenberg and Elman, Negishi and Eimas. *Science*, 284:436–37, 1999b.

Gary F. Marcus. Reply to Christiansen and Curtin. *Trends in Cognitive Sciences*, 3(8):290 – 291, 1999c.

Gary F. Marcus. Reply to Seidenberg and Elman. *Trends in Cognitive Sciences*, 3(8):288, 1999d.

Gary F. Marcus. Response to Technical Comment on Rule learning by seven-month-old infants and neural networks by gerry t.m. altmann and Zoltán Dienes. *Science*, 284(5416):875–876, 1999e.

J. L. McClelland and David C. Plaut. Does generalization in infant learning implicate abstract algebra-like rules? *Trends in Cognitive Sciences*, 3(5):166–168, 1999.

Steven Pinker. *Words and rules: The ingredients of language*. Basic Books, 2015.

Steven Pinker and Alan Prince. On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition*, 28(1-2):73–193, 1988.

Kim Plunkett and Virginia Marchman. From rote learning to system building: Acquiring verb morphology in children and connectionist nets. *Cognition*, 48(1):21–69, 1993.

D.E. Rumelhart and J.L. McClelland. On learning past tenses of English verbs. In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing, Vol. 2*, pages 318–362. MIT Press, Cambridge, MA, 1986.

Mark S. Seidenberg and Jeffrey L. Elman. Do infants learn grammar with algebra or statistics? *Science*, 284 (5413):433, 1999a.

Mark S. Seidenberg and Jeffrey L. Elman. Networks are not 'hidden rules'. *Trends in Cognitive Sciences*, 3(8): 288–289, 1999b.

Thomas R. Shultz. Rule learning by habituation can be simulated in neural networks. In *Proceedings of the twenty first annual conference of the Cognitive Science Society*, pages 665–670, 1999.

Thomas R. Shultz and Alan C. Bale. Neural network simulation of infant familiarization to artificial sentences: Rule-like behavior without explicit rules and variables. *Infancy*, 2(4):501–536, 2001.

Sylvian Sirois, David Buckingham, and Thomas R. Shultz. Artificial grammer learning by infants: an auto-associator perspective. *Developmental Science*, 3(4):442–456, 2000.

Analogy-based Reasoning With Memory Networks for Future Prediction

Daniel Andrade*
Data Science Research Laboratories
NEC Corporation, Japan
s-andrade@cj.jp.nec.com

Bing Bai
Department of Machine Learning
NEC Laboratories America
bbai@nec-labs.com

Ramkumar Rajendran†
Computer Science Department
Vanderbilt University
ramkumar.rajendran@vanderbilt.edu

Yotaro Watanabe
Data Science Research Laboratories
NEC Corporation, Japan
y-watanabe@fe.jp.nec.com

Abstract

Making predictions about what might happen in the future is important for reacting adequately in many situations. For example, observing that "Man kidnaps girl" may have the consequence that "Man kills girl". While this is part of common sense reasoning for humans, it is not obvious how machines can learn and generalize over such knowledge automatically. The order of event's textual occurrence in documents offers a clue to acquire such knowledge automatically. Here, we explore another clue, namely, logical and temporal relations of verbs from lexical resources. We argue that it is possible to generalize to unseen events, by using the entailment relation between two events expressed as (subject, verb, object) triples. We formulate our hypotheses of analogy-based reasoning for future prediction, and propose a memory network that incorporates our hypotheses. Our evaluation for predicting the next future event shows that the proposed model can be competitive to (deep) neural networks and rankSVM, while giving interpretable answers.

1 Introduction

Making predictions about what might happen in the future is important for reacting adequately in many situations. For example, observing that "Man kidnaps girl" may have the consequence that "Man kills girl". While this is part of common sense reasoning for humans, it is not obvious how machines can learn and generalize over such knowledge automatically.

One might think of learning such knowledge from massive amount of text data, such as news corpora. However, detecting temporal relations between events is still a difficult problem. Temporal order of events are often presented in different order in text. Although the problem can be partially addressed by using temporal markers like "afterwards", particularly with discourse parsers [18], overall, it remains a challenge.[3]

In this work, we propose to exploit the distinction between logical relations and temporal relations. We note that if an entailment relation holds between two events, then the second event is likely to be

*The first author is also associated with the Graduate University for Advanced Studies (SOKENDAI).

†The co-author contributed to this work while he was at NEC Corporation, Japan.

[3]For example, detecting implicit temporal relations (i.e. no temporal markers) is still a difficult problem for discourse parsers [18].

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

not a new future event.[4] For example, the phrase "man kissed woman" entails that "man met woman", where "man met woman" happens before (not after) "man kissed woman". To find such entailments, we can leverage relation of verbs in WordNet [5]. Verbs that tend to be in a temporal (happens-before) relation have been extracted on a large scale and are openly available in VerbOcean [4]. For example, we observe (subject, buy, object) tends to be temporally preceding (subject, use, object).

We present a model that can predict future events given a current event triplet (subject, verb, object). To make the model generalizable to unseen events, we adopt a deep learning structure such that the semantics of unseen events can be learned through word/event embeddings. We present a novel Memory Comparison Network (MCN) that can learn to compare and combine the similarity of input events to the event relations saved in memory. Our evaluation shows that this method is competitive to other (deep) neural networks and rankSVM [7], while giving interpretable answers.

In the first part of this work, in Section 2, we describe previous work related to future prediction. In Section 3, we discuss some connections between logical and temporal relations, and explain how we use lexical resources to create a knowledge base of positive and negative temporal relations. This knowledge base is then used by our experiments in the second part of our work.

In the second part, in Section 4, we formulate our assumptions of analogy based reasoning for future prediction. Underlying these assumptions, we propose our new method MCN. In Section 5, we describe several other methods that were previously proposed for future prediction, and ranking models that can be easily adapted to this task. In Section 6, we evaluate all methods on a future prediction task that requires to reason about unseen events. Finally, in Sections 7 and 8, we discuss some current limitations of our proposed method, and summarize our conclusions.

2 Related work

One line of research, pioneered by VerbOcean [4], extracts happens-before relations from large collections of texts using bootstrapping methods. In the context of script learning, corpora statistics, such as event bi-grams, are used to define a probability distribution over next possible future events [13, 3]. However, such models cannot generalize to situations of new events that have not been observed before. Therefore, the more recent methods proposed in [11, 15, 6] are based on word embeddings. Script learning is traditionally evaluated on small prototypical sequences that were manually created, or on event sequences that were automatically extracted from text. Due to the lack of training data, these models cannot learn to distinguish the fact that some events later in the text are actually entailed by events previously mentioned, i.e. already known events and new events are not distinguished.

3 Exploiting lexical resources

Our main focus is on distinguishing future events from other events. In texts, like news stories, an event e_l is more likely to have happened before event e_r (temporal order), if e_l occurs earlier in the text than e_r (textual order). However, there are also many situations where this is not the case: re-phrasing, introducing background knowledge, conclusions, etc. One obvious solution are discourse parsers. However, without explicit temporal markers, they suffer from low recall [18], and therefore in practice most script-learning systems use textual order as a proxy for temporal order. Here we explore whether common knowledge can help to improve future detection from event sequences in textual order.

We assume common knowledge is given in the form of simple relations (or rules) like

$$\text{(company, buy, share)} \rightarrow \text{(company, use, share)} ,$$

where "$\rightarrow$" denotes the temporal happens-before relation. In contrast, we denote the logical entailment (implication) relation by "$\Rightarrow$".

To extract such common knowledge rules we explore the use of the lexical resources WordNet and VerbOcean. As also partly mentioned in [5], logical and temporal relations are not independent, but

[4] We consider here entailment and (logical) implication as equivalent. In particular, synonyms are considered to be in an entailment relation, as in contrast to the classification by WordNet.

Table 1: Examples of several temporal and logical relations (relation types are shown in numbers relating to Figure 1).

Examples
(1) "minister leaves factory", "minister enters factory"
(2) "company donates money", "company gives money"
(3) "John starts marathon" , "John finishes marathon"
(4) "governor kisses girlfriend", "governor meets girlfriend"
(5) "people buy apple", "people use apple"
(6) "minister likes criticism", "minister hates criticism"
(7) "X's share falls 10%", "X's share rises 10%"

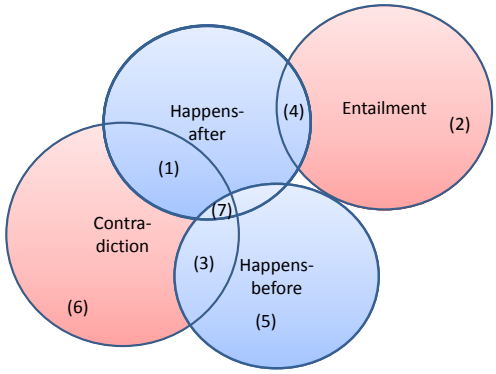


Figure 1: Illustration of logical (entailment, contradiction) and temporal (happens-before, happens-after) relation types. Examples are shown in Table 1.

an interesting overlap exists as illustrated in Figure 1, and corresponding examples shown in Table 1. We emphasis that, for temporal relations, the situation is not always as clear cut as shown in Figure 1 (e.g. repeated actions). Nevertheless, there is a tendency of event relations belonging mostly only to one relation. In particular, in the following, we consider "wrong" happens-before relations, as less likely to be true than "correct" happens-before relations.

3.1 Data creation

For simplicity, we restrict our investigation here to events of the form (subject, verb, object). All events are extracted from around 790k news articles in Reuters [9]. We preprocessed the English Reuters articles using the Stanford dependency parser and co-reference resolution [10]. We lemmatized all words, and for subjects and objects we considered only the head words, and ignored words like WH-pronouns.

All relations are defined between two events of the form (S, V_l, O) and (S, V_r, O), where subject S and object O are the same. As candidates we consider only events in sequence (occurrence in text).

Positive Samples We extract positive samples of the form $(S, V_l, O) \rightarrow (S, V_r^{pos}, O)$, if

1. $V_l \rightarrow V_r^{pos}$ is listed in VerbOcean as happens-before relation.
2. $\neg[V_l \Rightarrow V_r^{pos}]$ according to WordNet. That means, for example, if (S, V_r, O) is paraphrasing (S, V_l, O), then this is not considered as a temporal relation.

This way, we were able to extract 1699 positive samples. Examples are shown in Table 2.

Negative Samples Using VerbOcean, we extracted negative samples of the form $(S, V_l, O) \nrightarrow (S, V_r^{neg}, O)$, i.e. the event on the left hand (S, V_l, O) is the same as for a positive sample.[5] This way, we extracted 1177 negative samples.

[5] If $(S, V_l, O) \nrightarrow (S, V_r^{neg}, O)$, then $V_l \rightarrow V_r^{neg}$ is not listed in VerbOcean.

Table 2: Examples of happens-before relations extracted from news articles.

Examples
(company, buy, share) $\rightarrow$ (company, use, share)
(ex-husband, stalk, her) $\rightarrow$ (ex-husband, kill, her)
(farmer, plant, acre) $\rightarrow$ (farmer, harvest, acre)

There are several reasons for a relation not being in a temporal relation. Using VerbOcean and WordNet we analyzed the negative samples, and found that the majority (1030 relations) could not be classified with either VerbOcean or WordNet. We estimated conservatively that around 27% of these relations are false negatives: for a sub-set of 100 relations, we labeled a sample as a false negative, if it can have an interpretation as a happens-before relation.[6]

To simplify the task, we created a balanced data set, by pairing all positive and negative samples: each sample pair contains one positive and one negative sample, and the task is to find that the positive sample is more likely to be a happens-before relation than a negative sample. The resulting data set contains in total 1765 pairs.

4 Analogy-based reasoning for happens-before relation scoring

In the following, let r be a happens-before relation of the form:

$$r : e_l \rightarrow e_r \,,$$

where e_l and e_r are two events of the form (S, V_l, O) and (S, V_r, O), respectively. Furthermore, let e' be any event of the form (S', V', O').

Our working hypotheses consists of the following two claims:

$$\text{(I) If } \; (e' \Rightarrow e_l) \wedge (e_l \rightarrow e_r), \text{ then } \; e' \rightarrow e_r \,.$$
$$\text{(II) If } \; (e' \Rightarrow e_r) \wedge (e_l \rightarrow e_r), \text{ then } \; e_l \rightarrow e' \,.$$

For example, consider

"John buys computer" $\Rightarrow$ "John acquires computer" ,
"John acquires computer" $\rightarrow$ "John uses computer" .

Using (I), we can reason that:

"John buys computer" $\rightarrow$ "John uses computer" .

We note that, in some cases, " $\Rightarrow$" in (I) and (II) cannot be replace by " $\Leftarrow$". This is illustrated by the following example:

"John knows Sara" $\Leftarrow$ "John marries Sara" ,
"John marries Sara" $\rightarrow$ "John divorces from Sara" .

However, the next statement is considered wrong (or less likely to be true):

"John knows Sara" $\rightarrow$ "John divorces from Sara" .

In practice, using word embeddings, it can be difficult to distinguish between " $\Rightarrow$" and " $\Leftarrow$". Therefore, our proposed method uses the following simplified assumptions:

$$\text{(I}^*\text{) If } \; (e' \sim e_l) \wedge (e_l \rightarrow e_r), \text{ then } \; e' \rightarrow e_r \,.$$
$$\text{(II}^*\text{) If } \; (e' \sim e_r) \wedge (e_l \rightarrow e_r), \text{ then } \; e_l \rightarrow e' \,.$$

where $\sim$ denotes some similarity that can be measured by means of word embeddings.

[6] Therefore, this over-estimates the number of false negatives. This is because it also counts a happens-before relation that is less likely than a happens-after relation as a false negative.

4.1 Memory Comparison Network

We propose a memory-based network model that uses the assumptions (I*) and (II*). It bases its decision on one (or more) training samples that are similar to a test sample. In contrast to other methods like neural networks for script learning, and (non-linear) SVM ranking models, it has the advantage of giving an explanation of why a relation is considered (or not considered) as a happens-before relation.

In the following, let r_1 and r_2 be two happens-before relations of the form:

$$r_1 : (S_1, V_{l_1}, O_1) \rightarrow (S_1, V_{r_1}, O_1)\,,$$
$$r_2 : (S_2, V_{l_2}, O_2) \rightarrow (S_2, V_{r_2}, O_2)\,.$$

Let $\mathbf{x}_{s_i}$, $\mathbf{x}_{vli}$, $\mathbf{x}_{v_{r_i}}$ and $\mathbf{x}_{o_i} \in \mathbb{R}^d$ denote the word embeddings corresponding to S_i, V_{l_i}, V_{r_i} and O_i.[7]

We define the similarity between two relations r_1 and r_2 as:

$$sim_{\boldsymbol{\theta}}(r_1, r_2) = g_{\boldsymbol{\theta}}(\mathbf{x}_{v_{l_1}}^T \mathbf{x}_{v_{l_2}}) + g_{\boldsymbol{\theta}}(\mathbf{x}_{v_{r_1}}^T \mathbf{x}_{v_{r_2}})\,, \tag{1}$$

where $g_{\boldsymbol{\theta}}$ is an artificial neuron with $\boldsymbol{\theta} = \{\sigma, \beta\}$, a scale $\sigma \in \mathbb{R}$, and a bias $\beta \in \mathbb{R}$ parameter, followed by a non-linearity. We use as non-linearity the sigmoid function. Furthermore, here we assume that all word embeddings are l2-normalized.

Given the input relation $r : e_l \rightarrow e_r$, we test whether the relation is correct or wrong as follows. Let n_{pos} and n_{neg} denote the number of positive and negative training samples, respectively. First, we compare to all positive and negative training relations in the training data set, and denote the resulting vectors as $\boldsymbol{u}^{pos} \in \mathbb{R}^{n_{pos}}$ and $\boldsymbol{u}^{neg} \in \mathbb{R}^{n_{neg}}$, respectively. That is formally

$$u_t^{pos} = sim_{\boldsymbol{\theta}}(r, r_t^{pos}) \;\text{ and }\; u_t^{neg} = sim_{\boldsymbol{\theta}}(r, r_t^{neg})\,,$$

where r_t^{pos} and r_t^{neg} denotes the t-th positive/negative training sample.

Next, we define the score that r is correct/wrong as the weighted average of the relation similarities:

$$o^{pos} = \text{softmax}_\gamma(\mathbf{u}^{pos})^T \mathbf{u}^{pos} \;\text{ and }\; o^{neg} = \text{softmax}_\gamma(\mathbf{u}^{neg})^T \mathbf{u}^{neg} \tag{2}$$

where $\text{softmax}_\gamma(\mathbf{u})$ returns a column vector with the t-th output defined as

$$\text{softmax}_\gamma(\mathbf{u})_t = \frac{e^{\gamma u_t}}{\sum_i e^{\gamma u_i}}\,,$$

and $\gamma \in \mathbb{R}$ is a weighting parameter. Note that for $\gamma \rightarrow \infty$, $\text{softmax}_\gamma(\mathbf{u}) = \max(\mathbf{u})$, and for $\gamma = 0$, o is the average of $\mathbf{u}$.

Finally, we define the happens-before score for r as

$$l(e_l, e_r) = o^{pos}(e_l, e_r) - o^{neg}(e_l, e_r)\,. \tag{3}$$

The score $l(e_l, e_r)$ can be considered as an unnormalized log probability that relation r is a happens-before relation. The basic components of the network are illustrated in Figure 2.

For optimizing the parameters of our model we minimize the rank margin loss:

$$L(r^{pos}, r^{neg}) = \max\{0, 1 - l(e_l, e_r^{pos}) + l(e_l, e_r^{neg}))\}\,, \tag{4}$$

where $r^{pos} : e_l \rightarrow e_r^{pos}$ and $r^{neg} : e_l \rightarrow e_r^{neg}$ are positive and negative samples from the held-out training data. All parameters of the models are trained using stochastic gradient descent (SGD). Word embeddings ($\mathbf{x}_s, \mathbf{x}_v$, and $\mathbf{x}_o$) are kept fixed during training.

Our model can be interpreted as an instance of the Memory Networks proposed in [17]. Using the notation from [17], $I(\cdot)$ corresponds to the word embedding lookup, $G(\cdot)$ saves all training samples into the memory, the $O(\cdot)$ function corresponds to (o^{pos}, o^{neg}), and the output of $R(\cdot)$ equals Equation (3).

Our model also has similarity to the memory-based reasoning system proposed in [16], with two differences. First, we use here a trainable similarity measure, see Equation (1), rather than a fixed distance measure. Second, we use the trainable softmax_γ rather than max.

[7] Remark about our notation: we use bold fonts, like $\mathbf{v}$ to denote a column vector; $\mathbf{v}^T$ to denote the transpose, and v_t to denote the t-th dimension of $\mathbf{v}$.

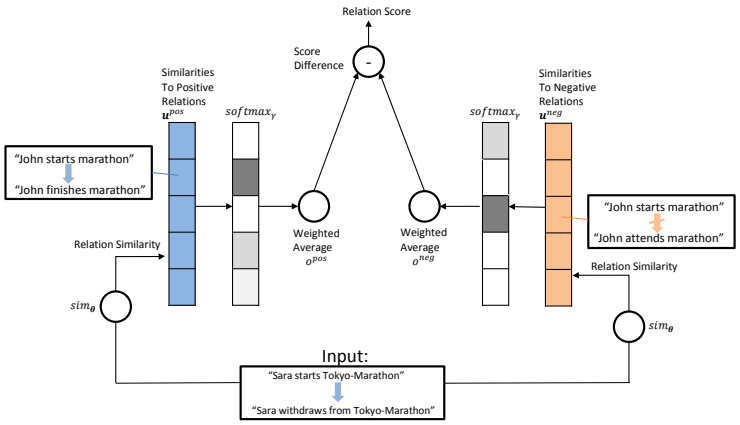


Figure 2: Illustration of proposed model.

5 Alternative ranking models

We also investigate several other models that can be applied for ranking temporal relations. All models that we consider are based on word embeddings in order to be able to generalize to unseen events.

Our first model is based on the bilinear model proposed in [1] for document retrieval, with scoring function $l(e_l, e_r) = \mathbf{z}_l^T M \mathbf{z}_r$, where $\mathbf{z}_l$ and $\mathbf{z}_r$ are the concatenated word embeddings $\mathbf{x}_s, \mathbf{x}_{v_l}, \mathbf{x}_o$ and $\mathbf{x}_s, \mathbf{x}_{v_r}, \mathbf{x}_o$, respectively, and parameter matrix $M \in \mathbb{R}^{3d \times 3d}$. We denote this model as Bai2009.

We also test three neural network architecture that were proposed in different contexts. The model in [2], originally proposed for semantic parsing, is a three layer network that can learn non-linear combinations of (subject, verb) and (verb, object) pairs. The non-linearity is achieved by the Hadamard product of the hidden layers. The original network can handle only events (relations between verb and objects, but not relations between events). We recursively extend the model to handle relations between events. We denote the model as Bordes2012.

In the context of script learning, recently two neural networks have been proposed for detecting happens-before relations. The model proposed in [11] (here denoted Modi2014) learns event embeddings parameterized with verb and context (subject or object) dependent embedding matrices. The event embeddings are then mapped to a score that indicates temporal time. To score a relation between events, we use the dot-product between the two events' embeddings.[8] The model in [6] suggests a deeper architecture than [11]. Their model (denoted here Granroth2016) uses additionally two non-linear layers for combining the left and right events. All neural networks and the Bai2009 model were trained in the same way as the proposed method, i.e. optimized with respect to rank margin loss using Equation (4).[9] For all methods, we kept the word embeddings fixed (i.e. no training), since this improved performance in general.

Our final two models use the rankSVM Algorithm proposed in [7] with the implementation from [8]. We tested both a linear and a rbf-kernel with the hyper-parameters optimized via grid-search. To represent a sample, we concatenate the embeddings of all words in the relation.

6 Experiments

We split the data set into training (around 50%), validation (around 25%), and testing (around 25%) set. Due to the relatively small size of the data we repeated each experiment 10 times for different random splits (training/validation/test).

[8] We also tried two variations: left and right events with different and same parameterization. However, the results did not change significantly.

[9] Originally, the model in [6] was optimized with respect to negative log-likelihood, however in our setting we found that rank-margin loss performed better.

Table 3: Mean accuracy and standard deviation (in brackets) of all methods for 10 random splits of training/validation/test.

Method	**Test Data**	**Validation Data**
Human estimate	76.7%	76.7%
Memory Comparison Network (softmax$_\gamma$, trained)	61.4 (11.5)	75.2 (7.6)
Granroth2016	60.9 (5.3)	72.7 (7.0)
Modi2014[10]	57.5 (7.8)	74.9 (6.6)
Bordes2012	58.3 (7.8)	74.9 (5.5)
Bai2009	58.9 (7.4)	72.7 (7.1)
rankSVM (rbf)	60.8 (6.1)	74.4 (6.7)
rankSVM (linear)	59.1 (9.4)	74.9 (6.5)
Random Baseline	50%	50%
Memory Comparison Network (softmax$_\gamma$, trained)	61.4 (11.5)	75.2 (7.6)
Memory Comparison Network (softmax$_\gamma$, initial parameters)	60.7 (5.9)	69.0 (9.7)
Memory Comparison Network (max, trained)	60.5 (6.2)	67.7 (9.7)
Memory Comparison Network (max, no parameters)	60.1 (5.8)	65.1 (10.9)
Memory Comparison Network (average, no parameters)	60.1 (5.9)	66.5 (8.7)

For the bilinear model and all neural networks, we performed up to 2000 epochs, and used early stopping with respect to the validation set. Some models were quite sensitive to the choice of the learning rates, so we tested 0.00001, 0.0001, and 0.001, and report the best results on the validation set.

For our proposed method, we set the learning rate constant to 0.001. Furthermore, we note that our proposed method requires two types of training data, one type of training data that is in memory, the other type that is used for learning the parameters. For the former and latter we used the training and validation fold, respectively. As initial parameters for this non-convex optimization problem we set $\sigma = 1.0, \beta = -0.5, \gamma = 5.0$, that were selected via the validation set.

For testing, we consider the challenging scenario, where the left event of the sample contains a verb that is not contained in the training set (and also not in the validation set).

We report accuracy, when asking the question: given observation (S, V_l, O), is (S, V_r^{pos}, O) more likely to be a future event than (S, V_r^{neg}, O)?

We used the 50 dimensional word embeddings from GloVe tool [12] trained on Wikipedia + Gigaword 5 provided by the authors (GloVe)[11].

The results of our method and previously proposed methods are shown in Table 3, upper half. By using the false-negative estimate from Section 3.1, we also calculated an estimate of the human performance on this task.[12]

The results suggest that our proposed model provides good generalization performance that is at par with the neural network recently proposed in [6] (Granroth2016), and SVM ranking with RBF-kernel. The results support our claim that the happens-before relation can be detected by analogy-based reasoning.

6.1 Analysis

We also compared to four variations of our proposed method. The results are shown in Table 3, lower half.

The first two variations use as similarity measure the addition of the word embeddings' inner products, i.e. $g_{\boldsymbol{\theta}}$ in Equation (1) is the identity function, and have no trainable parameters. The variation denoted by "Memory Comparison Network (max, no parameters)", is a kind of nearest neighbour ranking, that uses the max function instead of softmax$_\gamma$. The second variation, denoted by "Memory

[11] `http://nlp.stanford.edu/projects/glove/`

[12] We assume that distinguishing a false-negative from a true-positive is not possible (i.e. a human needs to guess), and that all guesses are wrong.

Table 4: Four examples with input relations, output scores and evidences by our proposed method.

input relation: (index,climb,percent) $\rightarrow$ (index,slide,percent)			
o^{pos}**:**	0.745	supporting evidence:	(rate,rise,percent) $\rightarrow$ (rate,tumble,percent)
o^{neg}**:**	0.697	supporting evidence:	(index,finish,point) $\nrightarrow$ (index,slide,point)
input relation: (parliament,discuss,budget) $\rightarrow$ (parliament,adopt,budget)			
o^{pos}**:**	0.412	supporting evidence:	(refiner,introduce,system) $\rightarrow$ (refiner,adopt,system)
o^{neg}**:**	0.352	supporting evidence:	(union,call,strike) $\nrightarrow$ (union,propose,strike)
input relation: (price,gain,cent) $\nrightarrow$ (price,strengthen,cent)			
o^{pos}**:**	0.542	supporting evidence:	(investment,build,plant) $\rightarrow$ (investment,expand,plant)
o^{neg}**:**	0.753	supporting evidence:	(dollar,rise,yen) $\nrightarrow$ (dollar,strengthen,yen)
input relation: (farmer,plant,acre) $\nrightarrow$ (farmer,seed,acre)			
o^{pos}**:**	0.136	supporting evidence:	(refinery,produce,tonne) $\rightarrow$ (refinery,process,tonne)
o^{neg}**:**	0.145	supporting evidence:	(refinery,produce,tonne) $\nrightarrow$ (refinery,receive,tonne)

Comparison Network (average, no parameters)", uses for o^{pos} and o^{neg}, in Equations (2), the average of $\mathbf{u}^{pos}$ and $\mathbf{u}^{neg}$, respectively. The performance of both variations is below our proposed method.

Furthermore, we compared to an alternative model, where the softmax$_\gamma$ is replaced by the max function, marked by "(max, trained)" in Table 3, lower half. Also, we compared to our proposed model, but without learning parameters, i.e. the parameters are set to the initial parameters, marked by "(softmax$_\gamma$, initial parameters)" in Table 3, lower half. We can see that the choice of softmax$_\gamma$, over max, improves performance, and that the training of all parameters with SGD is effective (in particular, see improvement on validation data).

Since our model uses analogy-based reasoning, we can easily identify "supporting evidence" for the output of our system. Four examples are shown in Table 4. Here, "supporting evidence" denotes the training sample with the highest similarity $sim_{\boldsymbol{\theta}}$ to the input. In the first and second example, the input is a happens-before relation, in the third and fourth example, the input is not a happens-before relation.[13]

7 Discussion

Our current method does not model the interaction between subject, object and verb. However, temporal relations can also crucially depend on subject and object. As an example, in our data set (see Table 2), we have the happens-before relation (company, buy, share) $\rightarrow$ (company, use, share). Clearly, if we replace the subject by "kid" and the object by "ice-cream", the happens-before relation becomes wrong, or much less likely. In particular, (kid, buy, ice-cream) $\rightarrow$ (kid, use, ice-cream) is much less likely than, for example, (kid, buy, ice-cream) $\rightarrow$ (kid, eat, ice-cream).[14]

Here, we compared two temporal rules r_1 and r_2 and asked which one is more likely, by ranking them. However, reasoning in terms of probabilities of future events, would allow us to integrate our predictions into a probabilistic reasoning framework like MLN [14].

8 Conclusions

We investigated how common knowledge, provided by lexical resources, can be generalized and used to predict future events. In particular, we proposed a memory network that can learn how to compare and combine the similarity of the input events to event relations saved in memory. This way our proposed method can generalize to unseen events and also provide evidence for its reasoning. Our experiments suggest that our method is competitive to other (deep) neural networks and rankSVM.

[13] Since we considered only the head, a unit like "percent" means "x percent", where x is some number.

[14] Partly, this could be addressed by considering also the selectional preference of verbs like "eat" and "use".

References

[1] Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Olivier Chapelle, and Kilian Weinberger. Supervised semantic indexing. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 187–196. ACM, 2009.

[2] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135, 2012.

[3] Nathanael Chambers and Daniel Jurafsky. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, pages 789–797, 2008.

[4] Timothy Chklovski and Patrick Pantel. Verbocean: Mining the web for fine-grained semantic verb relations. In *EMNLP*, volume 4, pages 33–40, 2004.

[5] Christiane Fellbaum and George Miller. Wordnet: An electronic lexical database. MIT Press, 1998.

[6] Mark Granroth-Wilding and Clark. What happens next? Event prediction using a compositional neural network model. *AAAI*, 2016.

[7] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[8] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

[9] David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397, 2004.

[10] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60, 2014.

[11] Ashutosh Modi and Ivan Titov. Inducing neural models of script knowledge. In *CoNLL*, volume 14, pages 49–57, 2014.

[12] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods on Natural Language Processing*, pages 1532–43, 2014.

[13] Karl Pichotta and Raymond J Mooney. Statistical script learning with multi-argument events. In *EACL*, volume 14, pages 220–229, 2014.

[14] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[15] Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. Script induction as language modeling. In *EMNLP*, 2015.

[16] Craig Stanfill and David Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29(12):1213–1228, 1986.

[17] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *ICLR 2015*, 2015.

[18] Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi Prasad, Christopher Bryant, and Attapol T. Rutherford. The conll-2015 shared task on shallow discourse parsing. In *CoNLL*, page 2, 2015.

Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation

Iulian Vlad Serban[*○]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Tim Klinger[◇]
IBM Research
T. J. Watson Research Center,
Yorktown Heights,
NY, USA

Gerald Tesauro[◇]
IBM Research
T. J. Watson Research Center,
Yorktown Heights,
NY, USA

Kartik Talamadupula[◇]
IBM Research
T. J. Watson Research Center,
Yorktown Heights,
NY, USA

Bowen Zhou[◇]
IBM Research
T. J. Watson Research Center,
Yorktown Heights,
NY, USA

Yoshua Bengio[†○]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Aaron Courville[○]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Abstract

We introduce a new class of models called *multiresolution recurrent neural networks*, which explicitly model natural language generation at two levels of abstraction: a sequence of high-level coarse tokens (coarse sequences), and a sequence of natural language words (e.g. sentences). The models follow a hierarchical generation process. First, the coarse sequence is generated through an inference mechanism parametrized by a recurrent neural network. Then, the natural language sequence is generated through a recurrent neural network language model. In our experiments, we extract coarse sequences from raw text using syntactic parsers, which are designed to capture compositional structure and semantics. This enables training both recurrent neural networks using log-likelihood. We apply the multiresolution recurrent neural networks to dialogue response generation in the technical support domain and compare them with several competing models. The multiresolution recurrent neural networks outperform competing models by a substantial margin, achieving state-of-the-art results according to both a human evaluation study and automatic evaluation metrics. Furthermore, experiments show the proposed models generate more fluent, relevant and goal-oriented responses.

Introduction

Recurrent neural networks (RNNs) have recently shown excellent performance on tasks such as language modelling (Graves, 2012; Mikolov *et al.*, 2010), machine translation (Sutskever *et al.*, 2014; Cho *et al.*, 2014) and speech recognition (Hinton *et al.*, 2012). Inspired by these advances, researchers have started to investigate such models for dialogue applications (Ritter *et al.*, 2011; Lowe *et al.*, 2015; Sordoni *et al.*, 2015b; Shang *et al.*, 2015; Li *et al.*, 2016; Bordes and Weston, 2016; Serban *et al.*, 2016a). In particular, researchers have proposed sequence-to-sequence (Seq2Seq) models for dialogue response generation. In this task, the model must generate an appropriate response given a dialogue context (history). Although this framework differs from the well-established goal-oriented

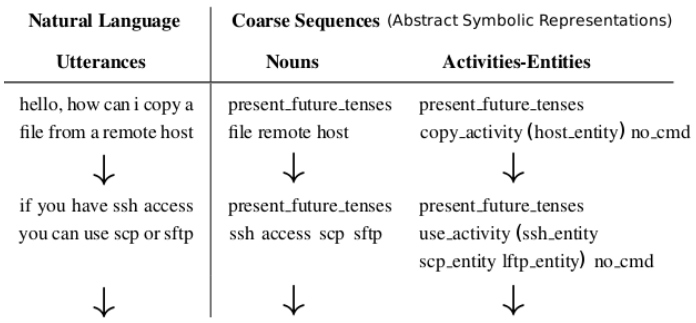


Figure 1: Example dialogue with coarse sequences. Arrows indicate turn change in the dialogue.

setting (Gorin *et al.*, 1997; Young, 2000; Singh *et al.*, 2002), these models have been applied to several real-world applications. One example is Microsoft's system Xiaolic, which now interacts with millions of users every day (Markoff and Mozur, 2015). Another example is Google's Smart Reply system (Kannan *et al.*, 2016). However, in spite of recent advances, current models cannot effectively incorporate dialogue context and generate meaningful, diverse on-topic responses (Li *et al.*, 2016).

To overcome these problems, we construct probabilistic models to *represent* and *reason* at several levels of abstraction. Explicitly representing multiple levels of abstraction should make it easier for models to remember and reason over long-term context, and to generate appropriate responses with compositional structure. For example, for technical support, this should help the models identify the user's problem and generate responses with appropriate, and even complex, technical instructions. Specifically, we propose a new class of models – called *multiresolution recurrent neural networks* (MrRNNs) – that operate on two parallel sequences: a sequence of high-level coarse tokens (coarse sequences), and a sequence of low-level natural language words (utterances). The coarse sequences are abstract symbolic representations of the utterances — analogous to the logical form of a sentence — and follow a stochastic process generated through an inference model. The inference model is parametrized by a recurrent neural network. The coarse sequences condition the natural language model through a hierarchical generation process. Given a dialogue context, the inference model first generates the next coarse sequence and then the natural language model uses the coarse sequence to generate the natural language response. This generation process adds high-level, compositional structure to the model, which helps generate meaningful and on-topic responses. In our experiments, the coarse sequences are defined either as noun sequences or activity-entity pairs (predicate-argument pairs) extracted by syntactic natural language parsers (see Figure 1). The syntactic parsing allows training MrRNNs using the joint log-likelihood over all observed sequences. We apply MrRNNs to dialogue response generation in the Ubuntu technical support domain, and compare them with recently proposed models. MrRNNs outperform competing approaches by a substantial margin according to both a human evaluation study and automatic evaluation metrics achieving a new state-of-the-art result. The results indicate MrRNNs hold significant potential for goal-oriented dialogue applications, as well as other applications involving natural language generation.

Model Architecture

Recurrent Neural Network Language Model

We first introduce the well-established recurrent neural network language model (RNNLM) (Mikolov *et al.*, 2010; Bengio *et al.*, 2003). Let $w_1, \ldots, w_N$ be a sequence of discrete variables, such as words in a document, called tokens where $w_n \in V$ for set (vocabulary) V^w. The RNNLM is a probabilistic generative model, with parameters θ, which decomposes the probability over tokens:

$$P_\theta(w_1, \ldots, w_N) = \prod_{n=1}^{N} P_\theta(w_n | w_1, \ldots, w_{n-1}). \quad (1)$$

where the output distribution uses a softmax RNN:

$$P_\theta(w_{n+1} = v|w_1, \ldots, w_n) = \frac{\exp(g(h_n, v))}{\sum_{v' \in V} \exp(g(h_n, v'))}, \tag{2}$$

$$h_n = f(h_{n-1}, w_n),\ g(h_n, v) = O_v^{\mathrm{T}} h_n\ \forall v \in V, \tag{3}$$

where f is the hidden state update function. We will assume it is either the LSTM or GRU gating unit (Hochreiter and Schmidhuber, 1997; Cho *et al.*, 2014).[1] The gating units may also be bidirectional. The matrix $I \in \mathbb{R}^{d_h \times |V|}$ is the input word embedding matrix, where row i contains the embedding for word index i and $d_h \in \mathbb{N}$ is the word embedding dimensionality. Similarly, the matrix $O \in \mathbb{R}^{d_h \times |V|}$ is the output word embedding matrix. According to the model, the probability of observing a token w at position $n+1$ increases if the context vector h_n has a higher dot-product with the word embedding corresponding to token w. The model parameters are usually learned by maximizing the log-likelihood (equivalent to minimizing the cross-entropy) on the training set using gradient descent.

Hierarchical Recurrent Encoder-Decoder

Our work builds upon the hierarchical recurrent encoder-decoder model (HRED) (Sordoni *et al.*, 2015a), which was previously proposed for dialogue (Serban *et al.*, 2016a). HRED decomposes a dialogue into a two-level hierarchy: a sequence of utterances, each of which is a sequence of words. Let $\mathbf{w}_1, \ldots, \mathbf{w}_N$ be a sequence of utterances with length N, where $\mathbf{w}_n = (w_{n,1}, \ldots, w_{n,K_n})$ is the n'th utterance consisting of K_n discrete tokens from vocabulary V^w. HRED decomposes the probability distribution into two products:

$$P_\theta(\mathbf{w}_1, \ldots, \mathbf{w}_N) = \prod_{n=1}^{N} P_\theta(\mathbf{w}_n|\mathbf{w}_{<n}) = \prod_{n=1}^{N} \prod_{m=1}^{K_n} P_\theta(w_{n,m}|w_{n,<m}, \mathbf{w}_{<n}). \tag{4}$$

This decomposition allows HRED to capture the hierarchical structure in natural language sequences, such as dialogue. HRED processes the utterances in three layers. First, each utterance sequence is encoded into a real-valued vector by an *encoder* RNN:

$$h^{\mathrm{e}}_{n,0} = \mathbf{0},\ \ h^{\mathrm{e}}_{n,m} = f^{\mathrm{e}}_\theta(h^{\mathrm{e}}_{n,m-1}, w_{n,m})\ \forall m = 1, \ldots, K_n,$$

where f^{e}_θ is the *encoder* RNN gating function. The last hidden state h^{e}_{n,K_n} summarizes the n'th utterance and is given to the higher-level *context* RNN:

$$h^{\mathrm{c}}_0 = \mathbf{0},\ \ h^{\mathrm{c}}_n = f^{\mathrm{c}}_\theta(h^{\mathrm{c}}_{n-1}, h^{\mathrm{e}}_{n,K_n})$$

where f^{c}_θ is the *context* RNN gating function taking two vectors as input. The *context* RNN acts like a memory module which remembers things over long time spans. Thus, the hidden state h^{c}_{n-1} summarizes the dialogue up to the $(n-1)$'th utterance. This is given to the *decoder* RNN:

$$h^{\mathrm{d}}_{n,0} = \mathbf{0},\ \ h^{\mathrm{d}}_{n,m} = f^{\mathrm{d}}_\theta(h^{\mathrm{d}}_{n,m-1},\ h^{\mathrm{c}}_{n-1}, w_{n,m})\ \forall m = 1, \ldots, K_n,$$

where f^{d}_θ is the LSTM gating function for the *decoder* RNN, which takes as input three vectors. The hidden state $h^{\mathrm{d}}_{n,m}$ is transformed through a one-layer neural network yielding the output vector $h^{\mathrm{o}}_{n,m}$. Finally, the output distribution is given by eq. (2) where h_n is replaced by $h^{\mathrm{o}}_{n,m}$.

Multiresolution RNN (MrRNN)

We now introduce the multiresolution recurrent neural network (MrRNN). As before, let $\mathbf{w}_1, \ldots, \mathbf{w}_N$ be a sequence of utterances of length N. Let $\mathbf{z}_1, \ldots, \mathbf{z}_N$ be the corresponding sequence of (high-level) *coarse* constituent sequences, also of length N, where $\mathbf{z}_n = (z_{n,1}, \ldots, z_{n,L_n})$ is the n'th constituent sequence consisting of L_n discrete tokens from vocabulary V^z. For example, each coarse sequence $\mathbf{z}_n$ could represent the nouns of the corresponding utterance $\mathbf{w}_n$. The coarse sequence $\mathbf{z}_n$ could also represent a symbolic abstraction of the corresponding utterance $\mathbf{w}_n$, such as the activity-entity pairs (predicate-argument pairs) discussed later (see Figure 1). The key assumption is that the coarse representations can be represented in a sequential structure.

The MrRNN is parametrized as a joint probabilistic model over the sequences $\mathbf{w}_1, \ldots, \mathbf{w}_N$ and $\mathbf{z}_1, \ldots, \mathbf{z}_N$. The MrRNN generates the n'th coarse sequence $\mathbf{z}_n$ by conditioning on the previous

[1] For the LSTM gating unit, we consider the hidden state h_m to be the input cell and memory cell concatenated.

coarse sequences $\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}$. Then, the model generates the n'th utterance $\mathbf{w}_n$ by conditioning on the previous utterances $\mathbf{w}_1, \ldots, \mathbf{w}_{n-1}$ and on the coarse sequences $\mathbf{z}_1, \ldots, \mathbf{z}_n$. Formally:

$$P_\theta(\mathbf{w}_1, \ldots, \mathbf{w}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = \prod_{n=1}^{N} P_\theta(\mathbf{z}_n|\mathbf{z}_{<n}) P_\theta(\mathbf{w}_n|\mathbf{w}_{<n}, \mathbf{z}_{<n}, \mathbf{z}_n), \tag{5}$$

where the conditional distributions decompose as:

$$P_\theta(\mathbf{z}_n|\mathbf{z}_{<n}) = \prod_{m=1}^{L_n} P_\theta(z_{n,m}|z_{n,<m}, \mathbf{z}_{<n}) \tag{6}$$

$$P_\theta(\mathbf{w}_n|\mathbf{w}_{<n}, \mathbf{z}_{<n}, \mathbf{z}_n) = \prod_{m=1}^{K_n} P_\theta(w_{n,m}|w_{n,<m}, \mathbf{w}_{<n}, \mathbf{z}_{<n}, \mathbf{z}_n) \tag{7}$$

The model differs from existing RNNLM models and Seq2Seq models, because it assigns probabilities to coarse sequence trajectories which condition the natural language utterances. We call the probability $P_\theta(\mathbf{z}_n|\mathbf{z}_{<n})$ for the *inference model*. Given the previous coarse sequences — such as the previous nouns in a dialogue or previous activity-entity pairs described later — the inference model is tasked with finding the most likely next coarse sequence trajectory. For example for technical support, if the noun *firefox* was observed in one coarse sequence, the inference model should assign a high probability to related nouns, such as *mozilla* and *browser*. Once the inference model has learned this, the model will be able to respond to a question like *"How do I install firefox?"* with an answer like *"You can download firefox from the mozilla website."*. Thus, the inference model is responsible for making the discrete high-level decisions (inference) about what to generate in natural language. The inference model computes the conditional distribution $P_\theta(\mathbf{z}_n|\mathbf{z}_{>n})$ as an HRED model, where the outputs are coarse sequences instead of natural language utterances.

The *natural language model* computes $P_\theta(\mathbf{w}_n|\mathbf{w}_{<n}, \mathbf{z}_{<n}, \mathbf{z}_n)$ as an HRED model applied to the utterances $\mathbf{w}_1, \ldots, \mathbf{w}_N$, but further conditioned on $\mathbf{z}_1, \ldots, \mathbf{z}_N$. It uses a GRU-gated *coarse prediction encoder* RNN, which encodes the coarse tokens $\mathbf{z}_1, \ldots, \mathbf{z}_n$:

$$h^{\text{p}}_{0,0} = \mathbf{0}, h^{\text{p}}_{n,0} = h^{\text{p}}_{n-1,L_{n-1}}$$
$$h^{\text{p}}_{n,m} = f^{\text{p}}_\theta(h^{\text{p}}_{n,m-1}, z_{n,m}) \; \forall m = 1, \ldots, L_n,$$

where f^{p}_θ is the gating function. All the coarse sequences up to the $(n-1)$'th utterance is encoded in the vector $h^{\text{p}}_{n-1,L_{n-1}}$. This encoding is different from the *context* RNN inside the inference model, because its task is to generate the next natural language utterance instead of the next coarse sequence. For this reason, the *coarse prediction encoder* RNN has separate word embedding parameters. The hidden state $h^{\text{p}}_{n-1,L_{n-1}}$ is given as input to the natural language *decoder* RNN – together with the hidden state of the natural language *context* RNN. As before, the output distribution over natural language words is given by eq. (2).

For training, a syntactic parser (described later) is used to extract the coarse sequences $\mathbf{z}_1, \ldots, \mathbf{z}_N$. This allows training all the model parameters w.r.t. the joint log-likelihood over both coarse sequences and natural language utterances. At test time, the model is given a dialogue context (dialogue history) and must generate an appropriate response. First, the coarse sequences corresponding to the context are extracted using the syntactic parser. Then, the inference model is executed to predict the most likely next coarse sequence using beam search. Next, the predicted coarse sequence is encoded by the *coarse prediction encoder* RNN. Finally, the natural language model generates the response using beam search based on the dialogue context and on the encoding of the predicted coarse sequence.

Experiments

We experiment on the task of response generation for dialogue. Given a dialogue context consisting of one or more utterances, the model must generate the next response in that dialogue. The specific task we consider is technical support for the Ubuntu operating system; the data we use is the Ubuntu Dialogue Corpus developed by (Lowe *et al.*, 2015). The corpus consists of about half a million natural language dialogues extracted from the *#Ubuntu* Internet Relayed Chat (IRC) channel. Users entering this chat channel usually have a specific technical problem. Typically, users first describe their problem, following which other users try to help them resolve it. The technical problems range from software-related issues (e.g. installing or upgrading existing software) and hardware-related issues (e.g. fixing broken drivers), to informational needs (e.g. finding software).

Coarse Sequence Representations

We develop two syntactic parsers for extracting the coarse sequences. The first is broadly applicable to a variety of dialogue tasks, and the second is specific to Ubuntu. The syntactic parsers are applied separately for each utterance in a dialogue. Figure 1 gives an example of extracted representations.

Noun Representation This parser is motivated by the observation that dialogues are topic-driven and that these topics may be characterized by nouns. The parser requires a part-of-speech tagger to identify the nouns in the dialogue and uses a set of 84 predefined stop words. It maps a natural language utterance to its corresponding coarse representation by extracting all the nouns using the part-of-speech tagger, and then removing all stop words and repeated words (keeping only the first occurrence of a given word). Utterances without nouns are assigned the "no_nouns" token. Finally, the parser adds the tense of each utterance to the beginning of the coarse sequence.

Activity-Entity Representation This parser is motivated by the observation that most dialogues are centered around *activities* and *entities*. For example, users frequently state a specific problem they want to resolve, e.g. *'how do I install program X?'* or *'My driver X doesn't work, how do I fix it?'* In response to such questions, other users often respond with specific instructions, e.g. *'Go to website X to download software Y'* or *'Try to execute command X'*. Thus, the principal information resides in the technical entities (e.g. *X*, *Y*) and in the verbs (e.g. *execute*, *download*). MrRNN should therefore be able to perform inference in the space of activities and entities. To extract these activities and entities, the parser uses a set of 192 activities (verbs) created by manual inspection, and a set of 3115 technical entities and 230 frequent terminal commands scraped from the web. The parser extracts the verbs and nouns from each natural language utterance, and maps them to the activity set and technical entity set. If an utterance does not contain any activity, the "none_activity" token is assigned. The parser also appends a binary variable to the end of the coarse representation indicating if a terminal command was detected in the utterance. Finally, the parser adds the tense of each utterance to the beginning of the coarse sequence.

Models

We implement all models in Theano. We train all models w.r.t. the log-likelihood or joint log-likelihood on the training set using Adam (Kingma and Ba, 2015). The models are trained using early stopping with patience based on the validation set log-likelihood. We choose model hyperparameters – such as the number of hidden units, word embedding dimensionality, and learning rate – based on the validation set log-likelihood. We use gradient clipping to stop the parameters from exploding (Pascanu *et al.*, 2012). We define the 20,000 most frequent words as the vocabulary, and map all other words to a special *unknown* token. Based on several experiments, we fix the word embedding dimensionality to size 300 for all models. We use a beam search of size 5 for generating the model responses.

Baseline Models We compare our models to several competing models proposed previously in the literature. The first baseline is the standard RNNLM with LSTM gating function (Mikolov *et al.*, 2010) (LSTM). This model is identical to the Seq2Seq LSTM model (Sutskever *et al.*, 2014), when the Seq2Seq LSTM model is trained to generate every utterance conditioned on all past utterances. The second baseline is HRED with LSTM gating function for the *decoder* RNN; and GRU gating function for the *encoder* RNN and *context* RNN. This model was developed specifically for dialogue response generation (Serban *et al.*, 2016a). The RNNLM model has 2000 hidden units with the LSTM gating function. The HRED model has 500, 1000, and 500 hidden units respectively for the *encoder* RNN, *context* RNN, and *decoder* RNN. The *encoder* RNN is bidirectional. The third baseline is the latent variable hierarchical recurrent encoder-decoder (VHRED), which extends HRED (Serban *et al.*, 2016b). We use the publicly available responses provided by Serban *et al.* (2016b).

In order to investigate the importance of the inference model, we introduce a fourth baseline, called *HRED + Activity-Entity Features*. This model simplifies MrRNN, such that the natural language *decoder* RNN is conditioned only on the *past* coarse tokens. In other words, this baseline has access to past activity-entity pairs – encoded using a GRU RNN – but it does not contain an inference model for predicting future coarse sequences. Thus, the baseline has access to the exact same information as the MrRNN. We use this baseline to evaluate the importance of the inference model. Based on the validation set log-likelihood, we specify the model architecture to have 500, 1000, and 2000 hidden units respectively for the *encoder*, *context* and *decoder* RNNs. The *encoder* RNN is bidirectional. The GRU RNN, which encodes the past coarse-level activity-entities sequences, has 500 hidden units.

MrRNN The inference model is parametrized as the HRED model (Serban *et al.*, 2016a) with 1000, 1000, and 2000 hidden units respectively for the coarse-level *encoder*, *context*, and *decoder* RNNs. The natural language model is parametrized as the HRED model with 500, 1000, and 2000 hidden units respectively for the natural language *encoder*, *context*, and *decoder* RNNs. The *coarse prediction encoder* GRU RNN has 500 hidden units. The *encoder* RNNs for the inference model and for the natural language model are both bidirectional.

Evaluation Methods

It has long been known that accurate evaluation of dialogue system responses is difficult (Schatzmann *et al.*, 2005). Liu *et al.* (2016) have recently shown that all automatic evaluation metrics adopted for such evaluation, including word overlap-based metrics such as BLEU and METEOR, have either very low or no correlation with human judgment of system performance. We therefore carry out an in-lab human study to evaluate the model responses. We recruit 5 human evaluators. We show each evaluator between 30 and 40 dialogue contexts with the ground truth response, and 4 candidate responses (HRED, HRED + Activity-Entity Features, MrRNN Noun, and MrRNN Activity-Entity). For each example, we ask the evaluators to compare the candidate responses to the ground truth response and dialogue context, and rate them for fluency and relevancy on a scale $0-4$: 0 means incomprehensible or no relevancy and 4 means flawless English or all relevant. This setup is similar to the evaluation setup used by Koehn and Monz (2006) and comparable to Liu et al. (2016).

We further propose a new set of evaluation metrics for the Ubuntu domain, in order to ease reproducibility and facilitate future evaluations. These metrics compare the activities and entities in the responses generated by the model with those of the ground truth responses. The assumption is that if a model generates responses with the same meaning as the ground truth responses – including expert responses, which often lead to solving the user's problem – then the model performs well. To compute the metrics, first the ground truth and model responses are mapped to their respective activity-entity representations using the syntactic parsing described earlier. Then, the overlap between their activities and entities are measured according to F1-score. Because the activities and entities reflect the principal information contained in a response – which are key to resolving the technical problem – these metrics should correlate well with solving user problems.

Table 1: Ubuntu evaluation using F1 metrics w.r.t. activities and entities on ground truth utterances (mean scores $\pm$ 90% confidence intervals), and human fluency and human relevancy scores given on a scale 0-4 (* indicates scores significantly different from baseline models at 90% confidence)

Model	Activity F1	Entity F1	Human Fluency	Human Relevancy
LSTM	1.18 ± 0.18	0.87 ± 0.15	-	-
HRED	4.34 ± 0.34	2.22 ± 0.25	2.98	1.01
VHRED	4.63 ± 0.34	2.53 ± 0.26	-	-
HRED + Act.-Ent. Features	5.46 ± 0.35	2.44 ± 0.26	2.96	0.75
MrRNN Noun	4.04 ± 0.33	$\mathbf{6.31 \pm 0.42}$	$\mathbf{3.48}^*$	$\mathbf{1.32}^*$
MrRNN Act.-Ent.	$\mathbf{11.43 \pm 0.54}$	3.72 ± 0.33	$\mathbf{3.42}^*$	1.04

Results

The results are given in Table 1. The MrRNNs perform substantially better than the baseline models w.r.t. both the human evaluation study and automatic evaluation metrics. MrRNN with noun representations obtains an entity F1 score at 6.31, while all baseline models obtain less than half between $0.87 - 2.53$. Further, human evaluators consistently rate its fluency and relevancy significantly higher than all the baseline models. MrRNN with activity representations obtains an activity F1 score at 11.43, while all other models obtain less than half F1 activity scores between $1.18 - 5.46$. It also performs substantially better than the baseline models w.r.t. the entity F1 score. In agreement with these results, human evaluators rate its fluency substantially higher than the baseline models. On the other hand, its relevancy is rated only slightly higher than HRED. We believe this effect is caused by the human evaluators focusing excessively on software entities in the dialogue responses – rather than focusing on on the actions/activities, which are critical to solving the actual technical problem posed by the user. In fact, our manual inspection of the generated responses confirm that its responses are more useful and goal-driven compared to the other model responses. Finally, the MrRNNs outperform the HRED baseline augmented activity-entity features across all metrics. This

Table 2: Model response examples. The arrows indicate change of turn.

Dialogue Context	Model Response	Predicted Coarse Sequence
Hello I tried to install ubuntu studio but I get W : Failed to fetch <url >404 Not Found when apt-get update → the ppa you added is not online	**MrRNN Act. -Ent.:** try a different mirror **MrRNN Noun:** something **VHRED:** but I dont know how to add it **HRED:** how do I do that ?	**MrRNN Act. -Ent.:** no_tenses try_activity no_cmd **MrRNN Noun:** past_present_tenses ubuntu studio <url>
hi community.. is there any difference between " /.bashrc " " /.bashrc. local " → yes, they're different files :-) the default /.bashrc sources /.bashrc. local → sorry I could undersatnd... If I write on terminal " gedit /.bashrc " it opens a file but how I open " /.bashrc. local " → " gedit /.bashrc. local "... → but it just open blank file nothing inside it	**MrRNN Act.-Ent.:** open up gedit /.bashrc, there is a file called " something " **MrRNN Noun:** the file is empty.. **VHRED:** it's in /.bashrc or /.bashrc **HRED:** what are you trying to do ?	**MrRNN Act.-Ent.:** present_tenses open_activity gedit_entity no_cmd **MrRNN Noun:** present_tenses community difference
Hey guys what do you in general use for irc something ubuntu xchat or xchat-gnome ? → without -gnome. that is just cut down → you mean drop xchat-gnome and go with xchat ?	**MrRNN Act. -Ent.:** im using xchat right now **MrRNN Noun:** what is xchat-gnome ? **VHRED:** correct **HRED:** yes	**MrRNN Act.-Ent.:** present_tenses use_activity xchat_entity no_cmd **MrRNN Noun:** present_tenses guys general irc ubuntu xchat xchatgnome
when setting up rules with iptables command only writes changes this file " etciptables. rules "? i ask this so i can backup before messing anything → sudo iptables-save something . dat to backup your rules restore with sudo iptables-restore < something . dat	**MrRNN Act. -Ent.:** I don't see any reason why you need iptables to do that **MrRNN Noun:** are you using ubuntu ? **VHRED:** thx **HRED:** thanks	**MrRNN Act.-Ent.:** present_tenses look_activity iptables_entity no_cmd **MrRNN Noun:** present_tenses rules iptables command file <path>

indicates that the inference model provides MrRNNs a critical advantage, by allowing the models to jointly represent and reason at multiple levels of abstraction in order to generate responses with high-level, compositional structure. Overall, the results indicate that the MrRNNs have learned to model high-level, goal-oriented sequential structure in the Ubuntu domain.

Model responses are shown in Table 2 together with the generated coarse responses. The MrRNN responses are generally very coherent and topic-oriented. In contrast, responses by the other baselines tend to be off-topic or very generic (Li *et al.*, 2016; Serban *et al.*, 2016a). This suggests that the hierarchical generation process and factorial representation of the coarse sequences help MrRNNs take dialogue context into account and generalize to new examples. In particular, MrRNN with activity-entity representation appears qualitatively to give more goal-oriented instructions in comparison to MrRNN with noun representation. This is also shown by the goal-oriented coarse sequences predicted by the inference model. This observation indicates that the hierarchical generation process helps MrRNN generate responses with more high-level, compositional structure.

Related Work

MrRNNs build upon the standard RNNLM (Mikolov *et al.*, 2010), which follows a sequential generation process where each word is generated conditioned on all previous words. The same applies to most Seq2Seq models for dialogue (Sordoni *et al.*, 2015b; Shang *et al.*, 2015; Li *et al.*, 2016; Serban *et al.*, 2016a) and other models involving natural language generation (Bahdanau *et al.*, 2015; Weston *et al.*, 2015; Graves *et al.*, 2014; Kumar *et al.*, 2016). Such models do not have any inference mechanism for predicting high-level structure in the response, and therefore have difficulty incorporating and generating high-level compositional structure.

MrRNNs are related to the VHRED model (Serban *et al.*, 2016b; Bowman *et al.*, 2015), which uses a continuous stochastic latent variable $\mathbf{z}_n$ for each utterance, conditioned on the previous utterances. VHRED generates an utterance by first sampling $\mathbf{z}_n$, and then generates the utterance $\mathbf{w}_n$ word by word. Similar to the MrRNNs, the prior distribution over the continuous latent variable $\mathbf{z}_n$ corresponds to an inference mechanism. However — unlike the sequential and discrete inference mechanism used by MrRNNs — the VHRED inference mechanism operates in a continuous space which cannot easily be mapped to discrete symbols (coarse tokens) or causal relations (e.g. that the token *mozilla* should follow the token *firefox*). The same differences apply to other neural network models with continuous latent variables, such as the Variational Recurrent Neural Network (Chung *et al.*, 2015), the Variational Recurrent Autoencoder (Fabius and van Amersfoort, 2014) and the Structured Variational Autoencoder (SVAE) (Johnson *et al.*, 2016). In addition to these, MrRNNs are related to the DrLM model proposed by Ji *et al.* (2016). This model uses one discrete stochastic latent variable (high-level token) z_n for each utterance, which is conditioned on the previous utterance. Although the discrete stochastic variables can be mapped to meaningful discrete symbols, they are

less expressive than the MrRNNs because there is only exactly one discrete variable per utterance and this variable depends only on the immediate preceding utterance. MrRNNs are also related to non-neural network latent variable models for dialogue, such as that by Zhai and Williams (2014) and by He *et al.* (2016). Learning task-specific and temporal abstractions for dialogue was also investigated in earlier work by Bangalore *et al.* (2008), by Crook *et al.* (2009) and others. MrRNNs contribute to this line of work by explicitly modeling the sequence of activities and entities.

From a symbolic representation and reasoning perspective, MrRNNs are related to Hierarchical Task Networks (HTNs) (Erol *et al.*, 1994), which have utilize a knowledge specification structure. Although HTNs have been used in the past as a search control strategy, HTNs also provide a separation of knowledge at different levels of granularity. In particular for dialogue, the RavenClaw system (Bohus and Rudnicky, 2003) applied this approach to separate the "task and discourse behavior specification". At test time, MrRNNs can be viewed as HTNs with a two-level hierarchy; conditioned on a given dialogue context, MrRNNs first find the high-level action (coarse sequence) and then find the low-level decomposition of that high-level action (natural language response).

Discussion

Current sequence-to-sequence models for dialogue response generation cannot effectively take dialogue context into account and cannot generate meaningful and diverse on-topic responses. To address these problems, we have proposed a new class of models called multiresolution recurrent neural networks (MrRNNs). MrRNNs model dialogue through a two-level hierarchical generation process over high-level coarse sequences and natural language utterances. The coarse sequences are generated through an inference model — parametrized by a recurrent neural network — and condition the natural language generation model through the hierarchical generation process. This process allows MrRNNs to *represent* and *reason* at several levels of abstraction. Combined with a syntactic parser, the MrRNNs are trained by optimizing the joint log-likelihood over the sequences at each level. Given a dialogue context, the inference model first generates the next coarse sequence and then the natural language model uses the coarse sequence to generate the natural language response. We have applied MrRNNs to dialogue response generation in the Ubuntu technical support domain. We have evaluated them through a human evaluation study and via automatic evaluation metrics. According to both human evaluators and automatic evaluation, MrRNNs improve substantially over competing models. By representing and reasoning about information at different levels of abstraction, and by jointly optimizing the generation process across abstraction levels, MrRNNs are able to generate more fluent, relevant and goal-oriented responses. The results indicate that it is not simply a matter of adding additional features for prediction – MrRNNs outperform a competitive model augmented with past coarse sequences as features – rather, it is the combination of representation and inference at multiple levels of abstraction that yields the improvements.

MrRNNs are a general framework for modeling discrete sequences at multiple levels of representation. We conjecture that they may be effective in broader natural language generation tasks, such as generating prose and persuasive argumentation, and other tasks involving discrete sequences, such as music composition.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bangalore, S., Di Fabbrizio, G., and Stent, A. (2008). Learning the structure of task-driven human–human dialogs. *IEEE Transactions on Audio, Speech, and Language Processing*, **16**(7), 1249–1259.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *JMLR*, **3**.

Bohus, D. and Rudnicky, A. I. (2003). Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda.

Bordes, A. and Weston, J. (2016). Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *arXiv:1511.06349*.

Cho, K. *et al.* (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. (2015). A recurrent latent variable model for sequential data. In *NIPS*, pages 2962–2970.

Crook, N., Granell, R., and Pulman, S. (2009). Unsupervised classification of dialogue acts using a dirichlet process mixture model. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics.

Erol, K., Hendler, J. A., and Nau, D. S. (1994). Umcp: A sound and complete procedure for hierarchical task-network planning. In *AIPS*, volume 94, pages 249–254.

Fabius, O. and van Amersfoort, J. R. (2014). Variational recurrent auto-encoders. *arXiv:1412.6581*.

Gorin, A. L., Riccardi, G., and Wright, J. H. (1997). How may i help you? *Speech communication*, **23**(1).

Graves, A. (2012). Sequence transduction with recurrent neural networks.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv:1410.5401*.

He, Z., Liu, X., Lv, P., and Wu, J. (2016). Hidden softmax sequence model for dialogue structure analysis. In *ACL*, pages 2063–2072.

Hinton, G. *et al.* (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, **29**(6), 82–97.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, **9**(8).

Ji, Y., Haffari, G., and Eisenstein, J. (2016). A latent variable recurrent neural network for discourse relation language models. In *NAACL-HLT*.

Johnson, M. J., Duvenaud, D., Wiltschko, A. B., Datta, S. R., and Adams, R. P. (2016). Structured VAEs: Composing probabilistic graphical models and variational autoencoders. *arXiv:1603.06277*.

Kannan, A., Kurach, K., Ravi, S., Kaufmann, T., Tomkins, A., Miklos, B., Corrado, G., Lukács, L., Ganea, M., Young, P., *et al.* (2016). Smart reply: Automated response suggestion for email. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, volume 36, pages 495–503.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Koehn, P. and Monz, C. (2006). Manual and automatic evaluation of machine translation between european languages. In *Workshop on Statistical Machine Translation, ACL*, pages 102–121.

Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. (2016). Ask me anything: Dynamic memory networks for natural language processing. *ICML*.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016). A diversity-promoting objective function for neural conversation models. In *NAACL*.

Liu, C.-W., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv:1603.08023*.

Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems. In *Proc. of SIGDIAL-2015*.

Markoff, J. and Mozur, P. (2015). For sympathetic ear, more chinese turn to smartphone program. *NY Times*.

Mikolov, T. *et al.* (2010). Recurrent neural network based language model. In *11th Proceedings of INTERSPEECH*, pages 1045–1048.

Pascanu, R., Mikolov, T., and Bengio, Y. (2012). On the difficulty of training recurrent neural networks. *Proceedings of the 30th International Conference on Machine Learning*, **28**.

Ritter, A., Cherry, C., and Dolan, W. B. (2011). Data-driven response generation in social media. In *EMNLP*.

Schatzmann, J., Georgila, K., and Young, S. (2005). Quantitative evaluation of user simulation techniques for spoken dialogue systems. In *6th SIGdial Workshop on DISCOURSE and DIALOGUE*.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016a). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, pages 3776–3784.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. (2016b). A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint arXiv:1605.06069*.

Shang, L., Lu, Z., and Li, H. (2015). Neural responding machine for short-text conversation. In *ACL-IJCNLP*.

Singh, S., Litman, D., Kearns, M., and Walker, M. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *JAIR*, **16**, 105–133.

Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Simonsen, J. G., and Nie, J.-Y. (2015a). A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Proc. of CIKM-2015*.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015b). A neural network approach to context-sensitive generation of conversational responses. In *NAACL*.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. *ICLR*.

Young, S. (2000). Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, **358**(1769).

Zhai, K. and Williams, J. D. (2014). Discovering latent structure in task-oriented dialogues. In *ACL*.

A SIMPLE BUT TOUGH-TO-BEAT BASELINE FOR SENTENCE EMBEDDINGS

Sanjeev Arora, Yingyu Liang, Tengyu Ma
Princeton University
{arora,yingyul,tengyu}@cs.princeton.edu

ABSTRACT

The success of neural network methods for computing word embeddings has motivated methods for generating semantic embeddings of longer pieces of text, such as sentences and paragraphs. Surprisingly, Wieting et al (ICLR'16) showed that such complicated methods are outperformed, especially in out-of-domain (transfer learning) settings, by simpler methods involving mild retraining of word embeddings and basic linear regression. The method of Wieting et al. requires retraining with a substantial labeled dataset such as Paraphrase Database (Ganitkevitch et al., 2013).

The current paper goes further, showing that the following completely unsupervised sentence embedding is a formidable baseline: Use word embeddings computed using one of the popular methods on unlabeled corpus like Wikipedia, represent the sentence by a weighted average of the word vectors, and then modify them a bit using PCA/SVD. This weighting improves performance by about 10% to 30% in textual similarity tasks, and beats sophisticated supervised methods including RNN's and LSTM's. It even improves Wieting et al.'s embeddings. This simple method should be used as the baseline to beat in future, especially when labeled training data is scarce or nonexistent.

The paper also gives a theoretical explanation of the success of the above unsupervised method using a latent variable generative model for sentences, which is a simple extension of the model in Arora et al. (TACL'16) with new "smoothing" terms that allow for words occurring out of context, as well as high probabilities for words like *and*, *not* in all contexts.

1 INTRODUCTION

Word embeddings computed using diverse methods are basic building blocks for Natural Language Processing (NLP) and Information Retrieval (IR). They capturing the similarities between words (e.g., (Bengio et al., 2003; Collobert & Weston, 2008; Mikolov et al., 2013a; Pennington et al., 2014)). Recent work has tried to compute embeddings that capture the semantics of word sequences (phrases, sentences, and paragraphs), with methods ranging from simple additional composition of the word vectors to sophisticated architectures such as convolutional neural networks and recurrent neural networks (e.g., (Iyyer et al., 2015; Le & Mikolov, 2014; Kiros et al., 2015; Socher et al., 2011; Blunsom et al., 2014; Tai et al., 2015; Wang et al., 2016)). Recently, (Wieting et al., 2016) learned general-purpose, paraphrastic sentence embeddings by starting with standard word embeddings and modifying them based on supervision from the Paraphrase pairs dataset (PPDB), and constructing sentence embeddings by training a simple word averaging model. This simple method leads to better performance on textual similarity tasks than a wide variety of methods and serves as a good initialization for textual classification tasks. However, supervision from the paraphrase dataset seems crucial, since they report that simple average of the initial word embeddings does not work very well.

Here we give a new sentence embedding method that is embarrassingly simple: just compute the weighted average of the word vectors in the sentence and then remove the projections of the average vectors on their first principal component ("*common component removal*"). Here the weight of a word w is $a/(a+p(w))$ with a being a parameter and $p(w)$ the (estimated) word frequency; we call this *smooth inverse frequency* (SIF). This method achieves significantly better performance than the

unweighted average on a variety of textual similarity tasks, and on most of these tasks even beats some sophisticated supervised methods tested in (Wieting et al., 2016), including some RNN and LSTM models. The method is well-suited for domain adaptation settings, i.e., word vectors trained on various kinds of corpora are used for computing the sentence embeddings in different testbeds. It is also fairly robust to the weighting scheme: using the word frequencies estimated from different corpora does not harm the performance; a wide range of the parameters a can achieve close-to-best results, and an even wider range can achieve significant improvement over unweighted average.

Of course, this SIF reweighting is highly reminiscent of TF-IDF reweighting from information retrieval (Sparck Jones, 1972; Robertson, 2004) if one treats a "sentence" as a "document" and make the reasonable assumption that the sentence doesn't typically contain repeated words. Such reweightings (or related ideas like removing frequent words from the vocabulary) are a good rule of thumb but has not had theoretical justification in a word embedding setting.

The current paper provides a theoretical justification for the reweighting using a generative model for sentences, which is a simple modification for the Random Walk on Discourses model for generating text in (Arora et al., 2016). In that paper, it was noted that the model theoretically implies a sentence embedding, namely, simple average of embeddings of all the words in it.

We modify this theoretical model, motivated by the empirical observation that most word embedding methods, since they seek to capture word cooccurence probabilities using vector inner product, end up giving large vectors to frequent words, as well as giving unnecessarily large inner products to word pairs, simply to fit the empirical observation that words sometimes occur out of context in documents. These anomalies cause the average of word vectors to have huge components along semantically meaningless directions. Our modification to the generative model of (Arora et al., 2016) allows "smoothing" terms, and then a max likelihood calculation leads to our SIF reweighting.

Interestingly, this theoretically derived SIF does better (by a few percent points) than traditional TF-IDF in our setting. The method also improves the sentence embeddings of Wieting et al., as seen in Table 1. Finally, we discovered that —contrary to widespread belief—**Word2Vec**(CBOW) also does *not* use simple average of word vectors in the model, as misleadingly suggested by the usual expression $\Pr[w|w_1, w_2, \ldots, w_5] \propto \exp(v_w \cdot (\frac{1}{5}\sum_i v_{w_i}))$. A dig into the implementation shows it implicitly uses a weighted average of word vectors —again, different from TF-IDF— and this weighting turns out to be quite similar in effect to ours. (See Section 3.1.)

2 Related Work

Word embeddings. Word embedding methods represent words as continuous vectors in a low dimensional space which capture lexical and semantic properties of words. They can be obtained from the internal representations from neural network models of text (Bengio et al., 2003; Collobert & Weston, 2008; Mikolov et al., 2013a) or by low rank approximation of co-occurrence statistics (Deerwester et al., 1990; Pennington et al., 2014). The two approaches are known to be closely related (Levy & Goldberg, 2014; Hashimoto et al., 2016; Arora et al., 2016).

Our work is most directly related work to (Arora et al., 2016), which proposed a random walk model for generating words in the documents. Our sentence vector can be seen as approximate inference of the latent variables in their generative model.

Phrase/Sentence/Paragraph embeddings. Previous works have computed phrase or sentence embeddings by composing word embeddings using operations on vectors and matrices e.g., (Mitchell & Lapata, 2008; 2010; Blacoe & Lapata, 2012). They found that coordinate-wise multiplication of the vectors performed very well among the binary operations studied. Unweighted averaging is also found to do well in representing short phrases (Mikolov et al., 2013a). Another approach is recursive neural networks (RNNs) defined on the parse tree, trained with supervision (Socher et al., 2011) or without (Socher et al., 2014). Simple RNNs can be viewed as a special case where the parse tree is replaced by a simple linear chain. For example, the skip-gram model (Mikolov et al., 2013b) is extended to incorporate a latent vector for the sequence, or to treat the sequences rather than the word as basic units. In (Le & Mikolov, 2014) each paragraph was assumed to have a latent paragraph vector, which influences the distribution of the words in the paragraph. **Skip-thought** of (Kiros et al., 2015) tries to reconstruct the surrounding sentences from surrounded one and treats

the hidden parameters as their vector representations. RNNs using long short-term memory (LSTM) capture long-distance dependency and have also been used for modeling sentences (Tai et al., 2015). Other neural network structures include convolution neural networks, such as (Blunsom et al., 2014) that uses a dynamic pooling to handle input sentences of varying length and do well in sentiment prediction and classification tasks.

The directed inspiration for our work is (Wieting et al., 2016) which learned paraphrastic sentence embeddings by using simple word averaging and also updating standard word embeddings based on supervision from paraphrase pairs; the supervision being used for both initialization and training.

3 A Simple Method for Sentence Embedding

We briefly recall the latent variable generative model for text in (Arora et al., 2016). The model treats corpus generation as a dynamic process, where the t-th word is produced at step t. The process is driven by the random walk of a discourse vector $c_t \in \Re^d$. Each word w in the vocabulary has a vector in $\Re^d$ as well; these are latent variables of the model. The discourse vector represents "what is being talked about." The inner product between the discourse vector c_t and the (time-invariant) word vector v_w for word w captures the correlations between the discourse and the word. The probability of observing a word w at time t is given by a log-linear word production model from Mnih and Hinton:

$$\Pr[w \text{ emitted at time } t \mid c_t] \propto \exp\left(\langle c_t, v_w \rangle\right). \tag{1}$$

The discourse vector c_t does a slow random walk (meaning that c_{t+1} is obtained from c_t by adding a small random displacement vector), so that nearby words are generated under similar discourses. It was shown in (Arora et al., 2016) that under some reasonable assumptions this model generates behavior –in terms of word-word cooccurence probabilities—that fits empirical works like word2vec and Glove. The random walk model can be relaxed to allow occasional big jumps in c_t, since a simple calculation shows that they have negligible effect on cooccurrence probabilities of words. The word vectors computed using this model are reported to be similar to those from Glove and word2vec(CBOW).

Our improved Random Walk model. Clearly, it is tempting to define the sentence embedding as follows: given a sentence s, do a MAP estimate of the discourse vectors that govern this sentence. We note that we assume the discourse vector c_t doesn't change much while the words in the sentence were emitted, and thus we can replace for simplicity all the c_t's in the sentence s by a single discourse vector c_s. In the paper (Arora et al., 2016), it was shown that the MAP estimate of c_s is —up to multiplication by scalar—the average of the embeddings of the words in the sentence.

In this paper, towards more realistic modeling, we change the model (1) as follows. This model has two types of "smoothing term", which are meant to account for the fact that some words occur out of context, and that some frequent words (presumably "the", "and " etc.) appear often regardless of the discourse. We first introduce an additive term $\alpha p(w)$ in the log-linear model, where $p(w)$ is the unigram probability (in the entire corpus) of word and α is a scalar. This allows words to occur even if their vectors have very low inner products with c_s. Secondly, we introduce a common discourse vector $c_0 \in \Re^d$ which serves as a correction term for the most frequent discourse that is often related to syntax. (Other possible correction is left to future work.) It boosts the co-occurrence probability of words that have a high component along c_0.

Concretely, given the discourse vector c_s, the probability of a word w is emitted in the sentence s is modeled by,

$$\Pr[w \text{ emitted in sentence } s \mid c_s] = \alpha p(w) + (1-\alpha)\frac{\exp\left(\langle \tilde{c}_s, v_w \rangle\right)}{Z_{\tilde{c}_s}}, \tag{2}$$
$$\text{where } \tilde{c}_s = \beta c_0 + (1-\beta)c_s, \;\; c_0 \perp c_s$$

where α and β are scalar hyperparameters, and $Z_{\tilde{c}_s} = \sum_{w \in \mathcal{V}} \exp\left(\langle \tilde{c}_s, v_w \rangle\right)$ is the normalizing constant (the partition function). We see that the model allows a word w unrelated to the discourse c_s to be omitted for two reasons: a) by chance from the term $\alpha p(w)$; b) if w is correlated with the common discourse vector c_0.

Algorithm 1 Sentence Embedding

Input: Word embeddings $\{v_w : w \in \mathcal{V}\}$, a set of sentences $\mathcal{S}$, parameter a and estimated marginal probabilities $\{p(w) : w \in \mathcal{V}\}$ of the words.
Output: Sentence embeddings $\{v_s : s \in \mathcal{S}\}$
1: **for all** sentence s in $\mathcal{S}$ **do**
2: $\quad v_s \leftarrow \frac{1}{|s|} \sum_{w \in s} \frac{a}{a+p(w)} v_w$
3: **end for**
4: Compute the first principal component u of $\{v_s : s \in \mathcal{S}\}$
5: **for all** sentence s in $\mathcal{S}$ **do**
6: $\quad v_s \leftarrow v_s - uu^\top v_s$
7: **end for**

Computing the sentence embedding. The word embeddings yielded by our model are actually the same (up to rescaling) as those by model (1). (In fact the new model was discovered by our detecting the common component c_0 in existing embeddings.) The sentence embedding will be defined as the max likelihood estimate for the vector c_s that generated it. (In this case MLE is the same as MAP since the prior is uniform.) We borrow the key modeling assumption of (Arora et al., 2016), namely that the word v_w's are roughly uniformly dispersed, which implies that the partition function Z_c is roughly the same in all directions. So assume that $Z_{\tilde{c}_s}$ is roughly the same, say Z for all $\tilde{c}_s$. By the model (2) the likelihood for the sentence is

$$p[s \mid c_s] = \prod_{w \in s} p(w \mid c_s) = \prod_{w \in s} \left[\alpha p(w) + (1-\alpha) \frac{\exp\left(\langle v_w, \tilde{c}_s \rangle\right)}{Z} \right].$$

Let

$$f_w(\tilde{c}_s) = \log \left[\alpha p(w) + (1-\alpha) \frac{\exp\left(\langle v_w, \tilde{c}_s \rangle\right)}{Z} \right]$$

denote the log likelihood of sentence s. Then, by simple calculus we have,

$$\nabla f_w(\tilde{c}_s) = \frac{1}{\alpha p(w) + (1-\alpha) \exp\left(\langle v_w, \tilde{c}_s \rangle\right) / Z} \frac{1-\alpha}{Z} \exp\left(\langle v_w, \tilde{c}_s \rangle\right) v_w.$$

Then by Taylor expansion, we have,

$$\begin{aligned} f_w(\tilde{c}_s) &\approx f_w(0) + \nabla f_w(0)^\top \tilde{c}_s \\ &= \text{constant} + \frac{(1-\alpha)/(\alpha Z)}{p(w) + (1-\alpha)/(\alpha Z)} \langle v_w, \tilde{c}_s \rangle . \end{aligned}$$

Therefore, the maximum likelihood estimator for $\tilde{c}_s$ on the unit sphere (ignoring normalization) is approximately,[1]

$$\arg\max \sum_{w \in s} f_w(\tilde{c}_s) \propto \sum_{w \in s} \frac{a}{p(w)+a} v_w, \text{ where } a = \frac{1-\alpha}{\alpha Z}. \tag{3}$$

That is, the MLE is approximately a weighted average of the vectors of the words in the sentence. Note that for more frequent words w, the weight $a/(p(w)+a)$ is smaller, so this naturally leads to a down weighting of the frequent words.

To estimate c_s, we estimate the direction c_0 by computing the first principal component of $\tilde{c}_s$'s for a set of sentences. In other words, the final sentence embedding is obtained by subtracting the projection of $\tilde{c}_s$'s to their first principal component. This is summarized in Algorithm 1.

3.1 Connection to Subsampling Probabilities in word2vec

Word2vec (Mikolov et al., 2013b) uses a sub-sampling technique which downsamples word w with probability proportional to $1/\sqrt{p(w)}$ where $p(w)$ is the marginal probability of the word w. This

[1] Note that $\max_{c:\|c\|=1} C + \langle c, g \rangle = g/\|g\|$ for any constant C.

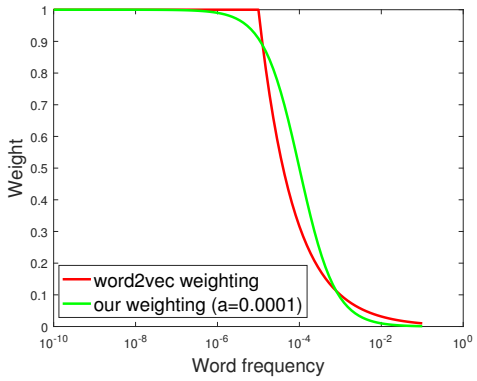


Figure 1: The subsampling probabilities in word2vec are similar to our weighting scheme.

heuristic not only speeds up the training but also learns more regular word representations. Here we explain that this corresponds to an implicit reweighting of the word vectors in the model and therefore the statistical benefit should be of no surprise.

Recall the vanilla CBOW model of word2vec:

$$\Pr[w_t \mid w_{t-1}, \ldots, w_{t-5}] \propto \exp\left(\langle \bar{v}_t, v_w \rangle\right), \text{ where } \bar{v}_t = \frac{1}{5}\sum_{i=1}^{5} v_{w_{t-i}}\,. \tag{4}$$

It can be shown that the loss (MLE) for the single word vector v_w (from this occurrence) can be abstractly written in the form,

$$g(v_w) = \gamma(\langle \bar{v}_t, v_w \rangle) + \text{negative sampling terms}\,,$$

where $\gamma(x) = \log(1/(1+e^{-x}))$ is the logistic function. Therefore, the gradient of $g(v_w)$ is

$$\nabla g(v_w) = \gamma'(\langle \bar{v}_t, v_w \rangle)\bar{v}_t = \alpha(v_{w_{t-5}} + v_{w_{t-4}} + v_{w_{t-3}} + v_{w_{t-2}} + v_{w_{t-1}})\,, \tag{5}$$

where α is a scalar. That is, *without* the sub-sampling trick, the update direction is the average of the word vectors in the window.

The sub-sampling trick in (Mikolov et al., 2013b) randomly selects the summands in equation (5) to "estimate" the gradient. Specifically, the sampled update direction is

$$\tilde{\nabla} g(v_w) = \alpha(J_5 v_{w_{t-5}} + J_4 v_{w_{t-4}} + J_3 v_{w_{t-3}} + J_2 v_{w_{t-2}} + J_1 v_{w_{t-1}}) \tag{6}$$

where J_k's are Bernoulli random variables with $\Pr[J_k = 1] = q(w_{t-k}) \triangleq \min\left\{1, \sqrt{\frac{10^{-5}}{p(w_{t-k})}}\right\}$. However, we note that $\tilde{\nabla} g(v_w)$ is (very) *biased* estimator! We have that the expectation of $\tilde{\nabla} g(v_w)$ is a *weighted* sum of the word vectors,

$$\mathbb{E}\left[\tilde{\nabla} g(v_w)\right] = \alpha(q(w_{t-5})v_{w_{t-5}} + q(w_{t-4})v_{w_{t-4}} + q(w_{t-3})v_{w_{t-3}} + q(w_{t-2})v_{w_{t-2}} + q(w_{t-1})v_{w_{t-1}})\,.$$

In fact, the expectation $\mathbb{E}[\tilde{\nabla} g(v_w)]$ corresponds to the gradient of a modified word2vec model with the average $\bar{v}_t$ (in equation (4)) being replaced by the weighted average $\sum_{k=1}^{5} q(w_{t-k})v_{w_{t-k}}$. Such a weighted model can also share the same form of what we derive form our random walk model as in equation (3). Moreover, the weighting $q(w_i)$ closely tracks our weighting scheme $a/(a + p(w))$ when using parameter $a = 10^{-4}$; see Figure 1 for an illustration. Therefore, the expected gradient here is approximately the estimated discourse vector in our model! Thus, word2vec with sub-sampling gradient heuristic corresponds to a stochastic gradient update method for using our weighting scheme.

	Results collected from (Wieting et al., 2016) except tfidf-GloVe											Our approach	
Supervised or not	Su.							Un.			Se.	Un.	Se.
Tasks	PP	PP -proj.	DAN	RNN	iRNN	LSTM (no)	LSTM (o.g.)	ST	avg-GloVe	tfidf-GloVe	avg-PSL	GloVe +WR	PSL +WR
STS'12	58.7	**60.0**	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	52.8	56.2	59.5
STS'13	55.8	56.8	54.2	44.7	56.7	45.2	41.5	24.8	42.3	52.1	46.4	56.6	**61.8**
STS'14	70.9	71.3	69.5	57.7	70.9	59.8	51.5	31.4	54.2	63.8	59.5	68.5	**73.5**
STS'15	75.8	74.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	60.0	71.7	**76.3**
SICK'14	71.6	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	66.4	72.2	**72.9**
Twitter'15	52.9	52.8	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	36.3	48.0	49.0

Table 1: Experimental results (Pearson's $r \times 100$) on textual similarity tasks. The highest score in each row is in boldface. The methods can be supervised (denoted as Su.), semi-supervised (Se.), or unsupervised (Un.). See the main text for the description of the methods.

4 EXPERIMENTS

4.1 TEXTUAL SIMILARITY TASKS

Datasets. We test our methods on the 22 textual similarity datasets including all the datasets from SemEval semantic textual similarity (STS) tasks (2012-2015) (Agirre et al., 2012; 2013; 2014; Agirrea et al., 2015), and the SemEval 2015 Twitter task (Xu et al., 2015) and the SemEval 2014 Semantic Relatedness task (Marelli et al., 2014). The objective of these tasks is to predict the similarity between two given sentences. The evaluation criterion is the Pearson's coefficient between the predicted scores and the ground-truth scores.

Experimental settings. We will compare our method with the following:

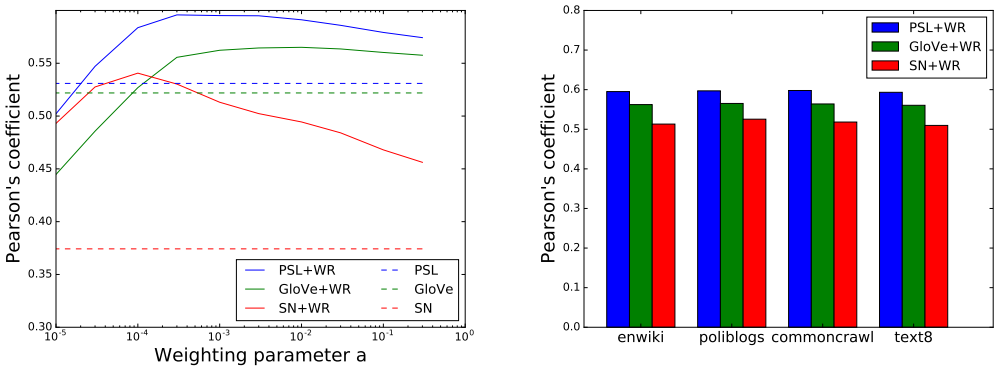
1. Unsupervised: ST, avg-GloVe, tfidf-GloVe. ST denotes the skip-thought vectors (Kiros et al., 2015), avg-GloVe denotes the unweighted average of the GloVe vectors (Pennington et al., 2014),[2] and tfidf-GloVe denotes the weighted average of GloVe vectors using TF-IDF weights.
2. Semi-supervised: avg-PSL. This method uses the unweighted average of the PARAGRAM-SL999 (PSL) word vectors from (Wieting et al., 2015). The word vectors are trained using labeled data, but the sentence embedding are computed by unweighted average without training.
3. Supervised: PP, PP-proj., DAN, RNN, iRNN, LSTM (o.g.), LSTM (no). All these methods are initialized with PSL word vectors and then trained on the PPDB dataset. PP and PP-proj. are proposed in (Wieting et al., 2016). The first is an average of the word vectors, and the second additionally adds a linear projection. The word vectors are updated during the training. DAN denotes the deep averaging network of (Iyyer et al., 2015). RNN denotes the classical recurrent neural network, and iRNN denotes a variant with the activation being the identity, and the weight matrices initialized to identity. The LSTM is the version from (Gers et al., 2002), either with output gates (denoted as LSTM(o.g.)) or without (denoted as LSTM (no)).

Our method can be applied to any types of word embeddings. To get a completely unsupervised method, we apply it to the GloVe vectors. The weighting parameter a is fixed to 10^{-3}, and the word frequencies $p(w)$ are estimated from the commoncrawl dataset.[3] This is denoted by GloVe+WR in Table 1. We also apply our method on the PSL vectors, denoted as PSL+WR, which is a semi-supervised method.

Results. The results are reported in Table 1. Each year there are 4 to 6 STS tasks. For clarity, we only report the average result for the STS tasks each year; the detailed results are in the appendix.

[2]We used the vectors that are publicly available at http://nlp.stanford.edu/projects/glove/. They are 300-dimensional vectors that were trained on the 840 billion token Common Crawl corpus.

[3]It is possible to tune the parameter a to get better results. The effect of a and the corpus for estimating word frequencies are studied in Section 4.1.1.



(a) (b)

Figure 2: Effect of weighting scheme in our method on the average performance on STS 2012 tasks. Best viewed in color. (a) Performance v.s. weighting parameter a. Three types of word vectors (PSL, GloVe, SN) are tested using $p(w)$ estimated on the enwiki dataset. The best performance is usually achieved at $a = 10^{-3}$ to $a = 10^{-4}$. (b) Performance v.s. datasets used for estimating $p(w)$. Four datasets (enwiki, poliblogs, commoncrawl, text8) are used to estimate $p(w)$ which is then used in our method. The parameter a is fixed to be 10^{-3}. The performance is almost the same for different settings.

The unsupervised method GloVe+WR improves upon avg-GloVe significantly by 10% to 30%, and beats the baselines by large margins. It achieves better performance than LSTM and RNN and comparable to DAN, even though the later three use supervision. This demonstrates the power of this simple method: it can be even stronger than highly-tuned supervisedly trained sophisticated models. Using TF-IDF weighting scheme also improves over the unweighted average, but not as much as our method.

The semi-supervised method PSL+WR achieves the best results for four out of the six tasks and are comparable to the best in the rest of two tasks. Overall, it outperforms the avg-PSL baseline and all the supervised models initialized with the same PSL vectors. This demonstrates the advantage of our method over the training for those models.

Finally, in the appendix, we showed that our two ideas all contribute to the improvement: for GloVe vectors, using smooth inverse frequency weighting alone improves over unweighted average by about 5%, using common component removal alone improves by 10%, and using both improves by 13%.

4.1.1 Effect of Weighting Parameter on Performance

We study the sensitivity of our method to the weighting parameter a, the method for computing word vectors, and the estimated word probabilities $p(w)$. First, we test the performance of three types of word vectors (PSL, GloVe, and SN) on the STS 2012 tasks. SN vectors are trained on the enwiki dataset (Wikimedia, 2012) using the method in (Arora et al., 2016), while PSL and GloVe vectors are those used in Table 1. We enumerate $a \in \{10^{-i}, 3 \times 10^{-i} : 1 \leq i \leq 5\}$ and use the $p(w)$ estimated on the enwiki dataset. Figure 2a shows that for all three kinds of word vectors, a wide range of a leads to significantly improved performance over the unweighted average. Best performance occurs from $a = 10^{-3}$ to $a = 10^{-4}$.

Next, we fix $a = 10^{-3}$ and use four very different datasets to estimate $p(w)$: enwiki (wikipedia, 3 billion tokens), poliblogs (Yano et al., 2009) (political blogs, 5 million), commoncrawl (Buck et al., 2014) (Internet crawl, 800 billion), text8 (Mahoney, 2008) (wiki subset, 1 million). Figure 2b shows performance is almost the same for all four settings.

	PP	DAN	RNN	LSTM (no)	LSTM (o.g.)	skip-thought	Ours
similarity (SICK)	84.9	85.96	73.13	85.45	83.41	85.8	**86.03**
entailment (SICK)	83.1	84.5	76.4	83.2	82.0	-	**84.6**
sentiment (SST)	79.4	83.4	86.5	86.6	**89.2**	-	82.2

Table 2: Results on similarity, entailment, and sentiment tasks. The sentence embeddings are computed unsupervisedly, and then used as features in downstream supervised tasks. The row for similarity (SICK) shows Pearson's $r \times 100$ and the last two rows show accuracy. The highest score in each row is in boldface. Results in Column 2 to 6 are collected from (Wieting et al., 2016), and those in Column 7 for skip-thought are from (Lei Ba et al., 2016).

The fact that our method can be applied on different types of word vectors trained on different corpora also suggests it should be useful across different domains. This is especially important for unsupervised methods, since the unlabeled data available may be collected in a different domain from the target application.

4.2 SUPERVISED TASKS

The sentence embeddings obtained by our method can be used as features for downstream supervised tasks. We consider three tasks: the SICK similarity task, the SICK entailment task, and the Stanford Sentiment Treebank (SST) binary classification task (Socher et al., 2013). To highlight the representation power of the sentence embeddings learned unsupervisedly, we fix the embeddings and only learn the classifier. Setup of supervised tasks mostly follow (Wieting et al., 2016) to allow fair comparison, i.e., the classifier a linear projection followed by the classifier in (Kiros et al., 2015). The linear projection maps the sentence embeddings into 2400 dimension (the same as the skip-thought vectors), and is learned during the training. We compare our method to PP, DAN, RNN, and LSTM, which are the methods used in Section 4.1. We also compare to the skip-thought vectors (with improved training in (Lei Ba et al., 2016)).

Results. Our method gets better or comparable performance compared to the competitors. It gets the best results for two of the tasks. This demonstrates the power of our simple method. We emphasize that our embeddings are unsupervisedly learned, while DAN, RNN, LSTM are trained with supervision. Furthermore, skip-thought vectors are much higher dimensional than ours (though projected into higher dimension, the original 300 dimensional embeddings contain all the information).

The advantage is not as significant as in the textual similarity tasks. This is possibly because similarity tasks rely directly upon cosine similarity, which favors our method's approach of removing the common components (which can be viewed as a form of denoising), while in supervised tasks, with the cost of some label information, the classifier can pick out the useful components and ignore the common ones.

5 CONCLUSIONS

This work provided a simple approach to sentence embedding, based on the discourse vectors in the random walk model for generating text (Arora et al., 2016). It is simple and unsupervised, but achieves significantly better performance than baselines on various textual similarity tasks, and can even beat sophisticated supervised methods such as some RNN and LSTM models. The sentence embeddings obtained can be used as features in downstream supervised tasks, which also leads to better or comparable results compared to the sophisticated methods.

REFERENCES

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pp. 385–393. Association for Computational Linguistics, 2012.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. Sem 2013 shared task: Semantic textual similarity. in second joint conference on lexical and computational semantics. In *Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, 2013.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pp. 81–91, 2014.

Eneko Agirrea, Carmen Baneab, Claire Cardiec, Daniel Cerd, Mona Diabe, Aitor Gonzalez-Agirrea, Weiwei Guof, Inigo Lopez-Gazpioa, Montse Maritxalara, Rada Mihalceab, et al. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pp. 252–263, 2015.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to PMI-based word embeddings. *Transaction of Association for Computational Linguistics*, 2016.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 2003.

William Blacoe and Mirella Lapata. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. N-gram counts and language models from the common crawl. In *Proceedings of the Language Resources and Evaluation Conference*, 2014.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 1990.

Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 2002.

Tatsunori B. Hashimoto, David Alvarez-Melis, and Tommi S. Jaakkola. Word embeddings as metric recovery in semantic spaces. *Transactions of the Association for Computational Linguistics*, 2016.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the Association for Computational Linguistics*, 2015.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, 2015.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning*, 2014.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *ArXiv e-prints*, 2016.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, 2014.

Matt Mahoney. Wikipedia text preprocess script. http://mattmahoney.net/dc/textdata.html, 2008. Accessed Mar-2015.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*, 2014.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 2013a.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2013b.

Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Association for Computational Linguistics*, 2008.

Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive science*, 2010.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme2, and Chris Callison-Burch. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2015.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing*, 2014.

Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation*, 2004.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, 2011.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, 2013.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2014.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Yashen Wang, Heyan Huang, Chong Feng, Qiang Zhou, Jiahui Gu, and Xiong Gao. Cse: Conceptual sentence embeddings based on attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 2015.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. In *International Conference on Learning Representations*, 2016.

Wikimedia. English Wikipedia dump. http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2, 2012. Accessed Mar-2015.

Wei Xu, Chris Callison-Burch, and William B Dolan. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). *Proceedings of SemEval*, 2015.

Tae Yano, William W Cohen, and Noah A Smith. Predicting response to political blog posts with topic models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2009.

A DETAILS OF EXPERIMENTAL SETTING

A.1 UNSUPERVISED TASK: TEXTUAL SIMILARITY

The competitors. We give a brief overview of the competitors. RNN is the classical recurrent neural network:

$$h_t = f(W_x W_w^{x_t} + W_h h_{t-1} + b)$$

where f is the activation, W_x, W_h and b are parameters, and x_t is the t-th token in the sentence. The sentence embedding of RNN is just the hidden vector of the last token. iRNN is a special RNN with the activation being the identity, the weight matrices initialized to identity, and b initialized to zero. LSTM (Hochreiter & Schmidhuber, 1997) is a recurrent neural network architecture designed to capture long-distance dependencies. Here, the version from (Gers et al., 2002) is used.

The supervised methods are initialized with PARAGRAM-SL999 (PSL) vectors, and trained using the approach of (Wieting et al., 2016) on the XL section of the PPDB data (Pavlick et al., 2015) which contains about 3 million unique phrase pairs. After training, the final models can be used to generate sentence embeddings on the test data. For hyperparameter tuning they used 100k examples sampled from PPDB XXL and trained for 5 epochs. Then after finding the hyperparameters that maximize Spearmans coefficients on the Pavlick et al. PPDB task, they are trained on the entire XL section of PPDB for 10 epochs. See (Wieting et al., 2016) and related papers for more details about these methods.

The tfidf-GloVe method is a weighted average of the GloVe vectors, where the weights are defined by the TF-IDF scheme. More precisely, the embedding of a sentence s is

$$v_s = \frac{1}{|s|} \sum_{w \in s} \text{IDF}_w v_w \tag{7}$$

where IDF_w is the inverse document frequency of w, and $|s|$ denotes the number of words in the sentence. Here, the TF part of the TF-IDF scheme is taken into account by the sum over $w \in s$. Furthermore, when computing IDF_w, each sentence is viewed as a "document":

$$\text{IDF}_w := \log \frac{1+N}{1+N_w}$$

where N is the total number of sentences and N_w is the number of sentences containing w, and 1 is added to avoid division by 0. In the experiments, we use all the textual similarity datasets to compute IDF_w.

Detailed experimental results. In the main body we present the average results for STS tasks by year. Each year there are actually 4 to 6 STS tasks, as shown in Table 3. Note that tasks with the same name in different years are actually different tasks. Here we provide the results for each tasks in Table 4. PSL+WR achieves the best results on 12 out of 22 tasks, PP and PP-proj. achieves on 3, tfidf-GloVe achieves on 2, and DAN, iRNN, and GloVe+WR achieves on 1. In general, our method improves the performance significantly compared to the unweighted average, though on rare cases such as MSRpar it can decrease the performance.

STS'12	STS'13	STS'14	STS'15
MSRpar	headline	deft forum	anwsers-forums
MSRvid	OnWN	deft news	answers-students
SMT-eur	FNWN	headline	belief
OnWN	SMT	images	headline
SMT-news		OnWN	images
		tweet news	

Table 3: The STS tasks by year. Note that tasks with the same name in different years are actually different tasks.

	Results collected from (Wieting et al., 2016) except tfidf-GloVe											Our approach	
Supervised or not	Su.							Un.			Se.	Un.	Se.
Tasks	PP	PP -proj.	DAN	RNN	iRNN	LSTM (no)	LSTM (o.g.)	ST	avg-GloVe	tfidf-GloVe	avg-PSL	GloVe +WR	PSL +WR
MSRpar	42.6	43.7	40.3	18.6	43.4	16.1	9.3	16.8	47.7	**50.3**	41.6	35.6	43.3
MSRvid	74.5	74.0	70.0	66.5	73.4	71.3	71.3	41.7	63.9	77.9	60.0	83.8	**84.1**
SMT-eur	47.3	49.4	43.8	40.9	47.1	41.8	44.3	35.2	46.0	**54.7**	42.4	49.9	44.8
OnWN	70.6	70.1	65.9	63.1	70.1	65.2	56.4	29.7	55.1	64.7	63.0	66.2	**71.8**
SMT-news	58.4	**62.8**	60.0	51.3	58.1	60.8	51.0	30.8	49.6	45.7	57.0	45.6	53.6
STS'12	58.7	**60.0**	56.0	48.1	58.4	51.0	46.4	30.8	52.5	58.7	52.8	56.2	59.5
headline	72.4	72.6	71.2	59.5	72.8	57.4	48.5	34.6	63.8	69.2	68.8	69.2	**74.1**
OnWN	67.7	68.0	64.1	54.6	69.4	68.5	50.4	10.0	49.0	72.9	48.0	**82.8**	82.0
FNWN	43.9	46.8	43.1	30.9	45.3	24.7	38.4	30.4	34.2	36.6	37.9	39.4	**52.4**
SMT	39.2	**39.8**	38.3	33.8	39.4	30.1	28.8	24.3	22.3	29.6	31.0	37.9	38.5
STS'13	55.8	56.8	54.2	44.7	56.7	45.2	41.5	24.8	42.3	52.1	46.4	56.6	**61.8**
deft forum	48.7	51.1	49.0	41.5	49.0	44.2	46.1	12.9	27.1	37.5	37.2	41.2	**51.4**
deft news	**73.1**	72.2	71.7	53.7	72.4	52.8	39.1	23.5	68.0	68.7	67.0	69.4	72.6
headline	69.7	**70.8**	69.2	57.5	70.2	57.5	50.9	37.8	59.5	63.7	65.3	64.7	70.1
images	78.5	78.1	76.9	67.6	78.2	68.5	62.9	51.2	61.0	72.5	62.0	82.6	**84.8**
OnWN	78.8	79.5	75.7	67.7	78.8	76.9	61.7	23.3	58.4	75.2	61.1	82.8	**84.5**
tweet news	76.4	75.8	74.2	58.0	76.9	58.7	48.2	39.9	51.2	65.1	64.7	70.1	**77.5**
STS'14	70.9	71.3	69.5	57.7	70.9	59.8	51.5	31.4	54.2	63.8	59.5	68.5	**73.5**
answers-forum	68.3	65.1	62.6	32.8	67.4	51.9	50.7	36.1	30.5	45.6	38.8	63.9	**70.1**
answers-student	**78.2**	77.8	78.1	64.7	**78.2**	71.5	55.7	33.0	63.0	63.9	69.2	70.4	75.9
belief	**76.2**	75.4	72.0	51.9	75.9	61.7	52.6	24.6	40.5	49.5	53.2	71.8	75.3
headline	74.8	75.2	73.5	65.3	75.1	64.0	56.6	43.6	61.8	70.9	69.0	70.7	**75.9**
images	81.4	80.3	77.5	71.4	81.1	70.4	64.2	17.7	67.5	72.9	69.9	81.5	**84.1**
STS'15	75.8	74.8	72.7	57.2	75.6	63.9	56.0	31.0	52.7	60.6	60.0	71.7	**76.3**
SICK'14	71.6	71.6	70.7	61.2	71.2	63.9	59.0	49.8	65.9	69.4	66.4	72.2	**72.9**
Twitter'15	52.9	52.8	**53.7**	45.1	52.9	47.6	36.1	24.7	30.3	33.8	36.3	48.0	49.0

Table 4: Experimental results (Pearson's $r \times 100$) on textual similarity tasks. The highest score in each row is in boldface. The methods can be supervised (denoted as Su.), semi-supervised (Se.), or unsupervised (Un.). See the main text for the description of the methods.

Effects of smooth inverse frequency and common component removal. There are two key ideas in our methods: smooth inverse frequency weighting (W) and common component removal (R). It is instructive to see their effects separately. Let GloVe+W denote the embeddings using only smooth inverse frequency, and GloVe+R denote that using only common component removal. Similarly define PSL+W and PSL+R. The results for these methods are shown in Table 5. When using GloVe vectors, W alone improves the performance of the unweighted average baseline by about 5%, R alone improves by 10%, W and R together improves by 13%. When using PSL vectors, W gets 10%, R gets 10%, W and R together gets 13%. In summary, both techniques are important for obtaining significant advantage over the unweighted average.

A.2 SUPERVISED TASKS

Setup of supervised tasks mostly follow (Wieting et al., 2016) to allow fair comparison: the sentence embeddings are fixed and fed into some classifier which are trained. For the SICK similarity task, given a pair of sentences with embeddings v_L and v_R, first do a linear projection:

$$h_L = W_p v_L, h_R = W_p v_R$$

where W_p is of size $300 \times d_p$ and is learned during training. $d_p = 2400$ matches the dimension of the skip-thought vectors. Then h_L and h_R are used in the objective function from (Tai et al., 2015). Almost the same approach is used for the entailment task.For the sentiment task, the classifier has a fully-connected layer with a sigmoid activation followed by a softmax layer.

Recall that our method has two steps: smooth inverse frequency weighting and common component removal. For the experiments here, we do not perform the common component removal, since it can be absorbed into the projection step. For the weighted average, the hyperparameter a is enumerated in $\{10^{-i}, 3 \times 10^{-i} : 2 \leq i \leq 3\}$. The other hyperparameters are enumerated as in (Wieting et al., 2016), and the same validation approach is used to select the final values.

	Unsupervised				Semi-supervised			
Tasks	avg-GloVe	GloVe+W	GloVe+R	GloVe+WR	avg-PSL	PSL+W	PSL+R	PSL+WR
MSRpar	47.7	43.6	36.4	35.6	41.6	40.9	42.5	43.3
MSRvid	63.9	78.7	79.4	83.8	60.0	80.4	76.4	84.1
SMT-eur	46.0	51.1	48.5	49.9	42.4	45.0	45.1	44.8
OnWN	55.1	54.3	68.3	66.2	63.0	67.8	71.0	71.8
SMT-news	49.6	42.2	45.6	45.6	57.0	56.2	50.7	53.6
STS'12	52.5	54.0	55.6	56.2	52.8	58.1	57.2	59.5
headline	63.8	63.8	68.9	69.2	68.8	72.6	72.7	74.1
OnWN	49.0	68.0	75.4	82.8	48.0	69.8	73.5	82.0
FNWN	34.2	23.0	34.9	39.4	37.9	49.3	40.7	52.4
SMT	22.3	29.5	36.4	37.9	31.0	39.2	37.3	38.5
STS'13	42.3	44.0	53.9	56.6	46.4	57.7	56.0	61.8
deft forum	27.1	29.1	39.8	41.2	37.2	45.8	45.3	51.4
deft news	68.0	68.5	66.6	69.4	67.0	75.1	67.4	72.6
headline	59.5	59.3	64.6	64.7	65.3	68.9	68.5	70.1
images	61.0	74.1	78.4	82.6	62.0	82.9	80.2	84.8
OnWN	58.4	68.0	77.6	82.8	61.1	77.6	77.7	84.5
tweet news	51.2	57.3	73.2	70.1	64.7	73.6	77.9	77.5
STS'14	54.2	59.4	66.7	68.5	59.5	70.7	69.5	73.5
answers-forum	30.5	41.4	58.4	63.9	38.8	56.0	61.0	70.1
answers-student	63.0	61.5	73.2	70.4	69.2	73.3	76.8	75.9
belief	40.5	47.7	69.5	71.8	53.2	64.3	71.3	75.3
headline	61.8	64.0	70.1	70.7	69.0	74.5	74.6	75.9
images	67.5	75.4	77.9	81.5	69.9	83.4	79.9	84.1
STS'15	52.7	58.0	69.8	71.7	60.0	70.3	72.7	76.3
SICK'14	65.9	70.5	70.6	72.2	66.4	73.1	70.3	72.9
Twitter'15	30.3	33.8	70.6	48.0	36.3	45.7	51.9	49.0

Table 5: Experimental results (Pearson's $r \times 100$) on textual similarity tasks using only smooth inverse frequency, using only common component removal, or using both.

Crossmodal language grounding, learning, and teaching

Stefan Heinrich, Cornelius Weber, Stefan Wermter
Department of Informatics, Knowledge Technology Group
University of Hamburg
Vogt-Koelln Str. 30, 22527 Hamburg, Germany
{heinrich,weber,wermter}@informatik.uni-hamburg.de

Ruobing Xie, Yankai Lin, Zhiyuan Liu
Department of Computer Science and Technology, Natural Language Processing Lab
Tsinghua University
Room 4-506, FIT Building, Beijing 100084, China
{liuzy}@tsinghua.edu.cn

Abstract

The human brain as one of the most complex dynamic systems enables us to communicate and externalise information by natural language. Despite extensive research, human-like communication with interactive robots is not yet possible, because we have not yet fully understood the mechanistic characteristics of the crossmodal binding between language, actions, and visual sensation that enable humans to acquire and use natural language. In this position paper we present vision- and action-embodied language learning research as part of a project investigating multi-modal learning. Our research endeavour includes to develop a) a cortical neural-network model that learns to ground language into crossmodal embodied perception and b) a knowledge-based teaching framework to bootstrap and scale-up the language acquisition to a level of language development in children of age up to two years. We embed this approach of internally grounding embodied experience and externally teaching abstract experience into the developmental robotics paradigm, by means of developing and employing a neuro-robot that is capable of multisensory perception and interaction. The proposed research contributes to designing neuroscientific experiments on discovering crossmodal integration particularly in language processing and to constructing future robotic companions capable of natural communication.

1 Introduction

While research in natural language processing has advanced in parsing and classifying large amounts of text, human-computer communication is still a major challenge: speech recognition is still limited to good signal-to-noise conditions or well adapted models; dialogue systems depend on a well-defined context; and interactive robots that match human communication performance are not yet available. One important reason is the fact that the crossmodal binding between language, actions, and visual events is not yet fully understood and realised in technical systems for the interaction with humans. Imaging techniques such as fMRI have provided a better understanding about which areas in the cortex are involved in natural language processing, and that these areas include somatosensory regions. Language studies have shown that there is a tight involvement of crossmodal sensation and action in speech processing and production as well as in language comprehension. Thus there is

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

increasing evidence that human language is embodied, which means that it is grounded in most if not all sensory and sensorimotor modalities, and that the human brain architecture favours the acquisition of language by means of crossmodal integration. However, while such language studies shed light on *what* information is processed *where*, they do not address *how* such areas function and compute.

In this position paper we present the vision-and action-embodied language learning research as part of the *Crossmodal Learning* (CML) project. In this project we aim to develop an integrated neural-network and knowledge-based model that processes auditory, visual and proprioceptive information and thus learns language by grounding speech in embodied perception. We develop the neural model, the neurobotic technology to embed it into a real child-like learning environment as well as the teaching framework, which substitutes a teaching caregiver. Such an endeavour will help to understand how information processing and learning takes place spatio-temporally in a cortical model with shared crossmodal representations. Overall, the model and teaching framework will provide a better understanding of how language is learned by grounding speech in embodied perception, how an instruction is interpreted to conduct actions in a real-word scene, and how learning can scale up to a human-like level of language competence.

2 A novel model for embodied language acquisition

Recent findings in neuroscience revealed evidence for embodied language processing in the brain. Specifically, Borghi et al. claimed that the sensorimotor system is involved during perception, action and language comprehension (3). In their review and meta-analysis they inferred that actions as well as words and sentences, which are referring to actions, are firstly encoded in terms of the overall goal (the overall concept) and then of the relevant effectors. In addition, Pulvermüller et al. suggested that for specific combinations of lexical and semantic information a combination of cortical areas, including auditory, motor, or olfactory cortices, can act as binding sites (13).

Latest cortical models in developmental robotics opened up the opportunity to observe language processing on humanoid neuro-robots (14; 5; 15), but could not yet achieve a plausibility and complexity with respect to these findings (6). Thus we cannot study the emergence of language and to learn how internal representations for the meaning of utterances develop, and how the temporal flow of information across the modalities is shaped (16). The goal of our research is to address these issues and develop a novel model based on crossmodal and multiple timescale characteristics.

2.1 How can language emerge in a computational model?

The central objectives in developing and studying a neurocognitively plausible model are to understand how language emerges and how the crossmodal integration bootstraps cognition. One specific research question is how temporal dynamic visual and proprioceptive perception contributes to the learning of the meaning of language production. The first hypothesis is that reverberating neural activities act as a memory in extracting a concept for the respective modality and joint modalities.

Secondly, we are keen to examine how the learning of meaning for holistic morphemes (or words) as well as for holo-phrases up to vocabulary-rich phrases scales up in an integrated architecture. The second hypothesis is that a certain degree of embodied grounding is necessary to learn a larger but hierarchically interdependent set of words and phrases. This means that a language learner requires a teaching environment that facilitates the learning of large and rich amounts of examples, including descriptions of possibly grounded or abstract interactions of a child-like learner with its environment (we will elaborate this in Sec. 3).

Validation of these concepts will require to embed the development of such a model as close as possible into a child-like learning environment. Constraints and assumptions can be clarified and limited by employing an embodied robotic implementation.

2.2 Crossmodal grounding in a cortical model

In previous models it was examined how the structure of the human cortex supports the integration of crossmodal input and how these integrative connection patterns result in high-level cognitive capabilities such as language understanding and production, while the development of specific connectivity is based on self-organization from the input data (8; 17). They showed that such a model

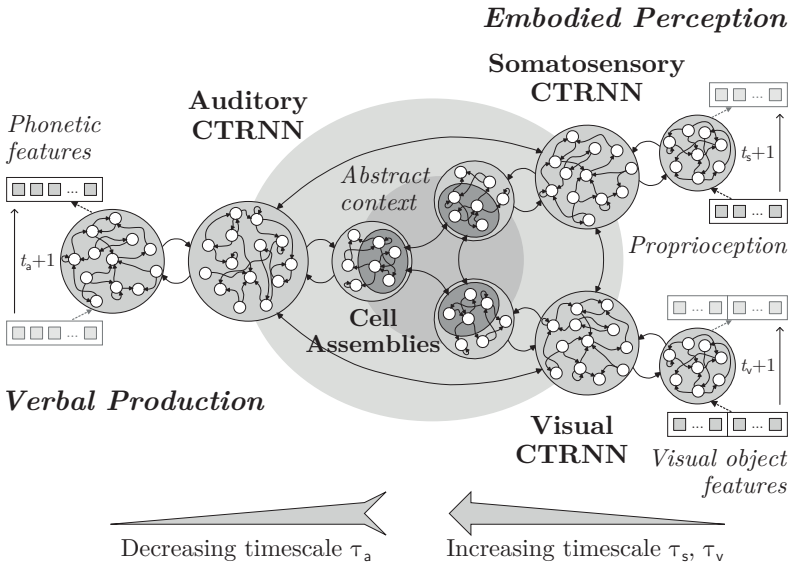


Figure 1: Architecture of the crossmodal model for grounding natural language, consisting of CTRNNs with multiple timescales for auditory production as well as for somatosensory and visual perception, and CAs for representing and processing the primitive concepts as well as abstract concepts. A sequence of phonemes (utterance) is produced over time, based on sequences of embodied crossmodal perception.

can learn sentences by extracting words and the underlying concepts from a verbal utterance and at the same time extract the concept from a body action both from the somatosensory as well as from the visual modalities (8; 9). The abstract concepts for somatosensory and visual perception are quite ambiguous on their own, but could be disambiguated in shared latent representations for the auditory production. To some extent, the model can generalise the verbal description to novel scenes where objects with different shape and colour properties can be recognised. In this paper we propose to develop the model further to allow for an up-scaling to reflect the language competence of up to two year-old children (6).

Our novel model is visualised in Fig. 1. As underlying characteristics, the model is based on a number of key principles:

a The architecture consists of several neurocognitively plausible *Continuous Time Recurrent Neural Networks* (CTRNNs).

b The layers are designed with varying leakage characteristics, thus operating on multiple timescales (19). This is inspired by the findings that a distinct increase in timescale is inherent along the caudal-rostral axis in the frontal lobe (1), indicating that layers of neurons in *higher* level areas process on slow dynamics and high abstraction, whereas neurons in sensory or motor areas process on fast dynamics.

c The layers are on the conceptual level interconnected in *Cell Assemblies* (CAs), thus exchanging and merging the respective auditory, somatosensory, and visual abstract representations and (4).

d Shortcut connections are included between intermediate layers in the respective modalities. Although in the brain the motor areas for speech output (controlling muscles for the lips, tongue and the larynx) and somatosensory areas as well as areas involved in visual perception may have different degrees of interconnectivity, the fusion of the information that these areas process happens in higher regions (13). In language processing this indicates that higher level concepts can form by the activation of large and highly distributed CAs of neurons that are strongly and reciprocally connected (11). Other CAs can take on the role of mediators between those concept-CAs and smaller CAs, which represent specific semantics like morphemes and lemmas.

3 A novel knowledgeable language teacher

A growing infant is exposed to many contextually varying discourses to pick up word-segmentation, word-object binding, and generalisation (6). Therefore, the training of a plausible model requires data that stems from some real embodied interactions and is available in large quantities.

To instruct the neural model for embodied language acquisition, we thus propose a novel knowledgeable language teaching system. This artificial teacher is considered to substitute a knowledgeable caregiver that instructs the crossmodal neural model embedded in a learning agent.

3.1 Central goal and functionality

The knowledge-based machine teaching system aims to automatically generate instances to facilitate language learning of the humanoid neuro-robot learner (we will elaborate this in Sec. 4) and must have access to crossmodal observations as well. Since language learning in humans is complex, and requires a child to experience several months of exposure to language, to multisensory context, and to active teaching by a caregiver to understand utterances, it would be extremely time-consuming and impractical for people to act as a language teacher to a robotic learner.

To simulate the real-world teaching scenario between parents and children, the teaching system will automatically provide a large number of crossmodal experiences, containing both possible interactions with objects and their verbal descriptions for neural model learning. In this teaching system, crossmodal information such as images, videos and describing texts are combined for a better overall learning from different aspects. Knowledge graphs are also considered as high-quality structured information for a deeper understanding of the attributes and relationships between objects. This scenario-focused knowledge accounts for core concepts (e.g. objects) and their attributes (e.g. actions afforded by the objects) that may be embodied for the robotic agent. Moreover, the teaching system is supposed to be able to efficiently repeat training sessions with fixed or varying conditions, and also help to avoid the experimenter's bias that is caused by human teachers.

3.2 Methodology

We will implement our crossmodal knowledge-based machine teaching system in a way that the teacher can observe the scene in which the learner interacts, and can describe the scene with a verbal description. The description can be both specific or constructive. Moreover, we will provide crossmodal knowledge in the form of both, visual actions (not necessarily containing embodied somatosensory cues) and verbal descriptions. More specifically, the construction of the teaching system contains four steps:

1. Collect crossmodal information,
2. Develop joint crossmodal knowledge representations,
3. Combine embodied experience with abstract experience of the knowledge-based system,
4. Provide efficient test cases for evaluation.

For the first step, we collect knowledge from a) video streams from the robotic learner interacting with the objects together with a human description of physical respective scene, and b) existing large-scale knowledge bases such as Freebase. Secondly, a joint crossmodal knowledge representation model will be developed to better understand potential attributes and relations between objects. The representation of images and video streams could be inspired by the huge success of convolutional neural networks for a variety of tasks like image classification (10), while the representation of knowledge could be learned under the state-of-the-art framework of translation-based methods (12). Moreover, we will also exploit several techniques (such as efficient object detection and entity recognition) to enrich video streams with text knowledge, to simulate further experience.

As the third step, we combine embodied interactions with abstract knowledge from the knowledge-based teaching system into the teaching process. The robotic learner must be able to bridge the gap from words and concepts learned from embodied interaction to additional knowledge learned from the teaching system. Thus, related examples will be provided that the learner has only experienced visually or somatosensorily, or may have never experienced at all. For example, the learner may experience a red cup and learn its attributes (both how a cup looks according to its shape and colour

and how it feels by touching). In addition, the teaching system will enable the robotic learner to extensively learn about various types of cups with different shapes and colors (or even teacups and bowls) from images or videos.

Finally, to evaluate whether the robotic learner can develop appropriate understandings and motor actions, the knowledge-based teacher should also provide efficient test cases. Since we simulate the real-world language teaching, the test cases will be embedded in a crossmodal scenario.

3.3 Significance and feasibility

Knowledge is essential and plays a significant role in the teaching system. First, we suppose that structured knowledge reflects the cognition to the real world. Cognition could compose discrete learned information to a structured knowledge graph, linking objects to each other or to their attributes, which helps language learning. Second, the joint knowledge representation enables association to similar objects (e.g. apple and orange) even if are never experienced by the learner. Knowledge inference is also available via the crossmodal representations, so that we can better understand more complicated instructions. Third, structured knowledge provides information for disambiguation, which is an essential challenge in language learning and understanding. It is natural to construct structured knowledge simultaneously while learning language.

Previous work on knowledge representation builds the foundation of our knowledge-based teaching system. TransE projects both, entities and relations, into the same continuous low-dimensional vector space, interpreting relations as translating operations between head and tail entities (2). This helps us to build relationships between entities, which improves the ability of association in language learning. As for crossmodal learning in knowledge representation, DKRL proposes a novel description-based representation for each entity (18), which combines textual information in entity descriptions with structured information in knowledge graphs. Our method of a crossmodal knowledge representation could be inspired by this framework. These recent approaches will form a basis to explore scenario-focused crossmodal knowledge extraction to enable a robot to learn languages.

4 Technologies and experimental design

Embodied language learning of the neurocognitive model will be studied on an interactive humanoid robot platform that is instructed by the knowledgeable teacher. As a first important characteristic, the learner must feature multimodal sensation capabilities to emulate the rich perception of a learning infant. For our particular scenario, the learner is supposed to interact with different objects in its field of view and must capture continuous stereo-visual and somatosensory perception – specifically proprioceptive information for an arm, but also tactile information for the hand – as well as perceive verbal descriptions from the knowledgeable teacher (see Fig. 2). The second important characteristic is the knowledgeable teacher that can observe and interpret the interactions and also inform about knowledge that is partially related to the interaction or even disconnected at all. The platform is overall related to the developmental robotics approach, in terms of capturing developmental principles and mechanisms observed in the natural cognitive systems of children (6), but is designed bottom-up from the neurocognitive model (8).

4.1 Evaluation and analysis methodology

To analyse the model's characteristics, we are interested to identify parameter settings for the best (relative) generalisation capabilities in order to analyse the information patterns that emerges for different parts of the architecture. Inspired from infant learning the evaluation will be embedded in the real world scenario, where the robot is supposed to show ideal performance for novel but related scenes, compared to the learned natural language interactions with the teacher and its environment (6). The analysis will be focused on correlating and quantifying the latent representations that might form in different higher level areas for specific modalities and in the higher level as well as mediating CAs between the modalities. This methodology is related to analysing representations from MEG or fMRI data (7), and can reveal how the architecture self-organises, based on the spatio-temporal dynamics as well as the hierarchical dependencies in the multi-modal training data.

Figure 2: The learner explores interaction with objects, while the knowledgeable teacher describes the respective interaction in natural language.

4.2 Up-scaling and meaningful uncertainty in a learner-caregiver scenario

In order to scale the model, the neuro-robotic learner will be exposed to rich permutations of temporal dynamic multimodal perception and desired auditory productions, thus to multi-variance data with a tremendous complexity and seemingly large degree of uncertainty. A central hypothesis for the learning is that a neurocognitively plausible model might be able to self-organise the binding of coherent sounds (or words) with visual entities or certain arm movements (primitive actions), but could heavily benefit from scaffolding: Similar to language learning in infants the learning could be shaped by teaching simple words and holo-phrases first, and then more complex utterances – without altering weights for certain layers in the architecture. In preliminary tests we learned that layers with a medium timescale show a predisposition in capturing holistic words and can be used to generate novel phrases. In addition, such step-by-step learning could also reveal differences and similarities in developed internal representations that emerge from learning.

5 Conclusion

We propose a research approach for understanding the temporal dynamics and mechanism underlying language processing and acquisition in the human brain. The development and study of our cortical neural model, integrated in a knowledgeable teaching framework, can provide theoretical and experimental evidence for the general hypothesis that a neural architecture that maximizes crossmodal as opposed to unimodal representations requires fewer resources and learning cycles, and can achieve better generalisation to unknown circumstances but also greater robustness in memorising and accessing its stored representations. With such an outcome we can design novel neuroscientific experiments on discovering crossmodal integration particularly in language processing and construct future robotic companions that provide better communication capabilities.

Acknowledgments

The authors gratefully acknowledge support from the German Research Foundation (DFG) and the National Science Foundation of China (NSFC) under project Crossmodal Learning, TRR-169.

References

[1] D. Badre, A. S. Kayser, and M. D'Esposito. Frontal cortex and the discovery of abstract action rules. *Neuron*, 66(2):315–326, 2010.

[2] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proc. NIPS 2013*, pages 2787–2795, 2013.

[3] A. M. Borghi, C. Gianelli, and C. Scorolli. Sentence comprehension: effectors and goals, self and others. An overview of experiments and implications for robotics. *Frontiers in Neurorobotics*, 4(3):8 p., 2010.

[4] V. Braitenberg and A. Schüz. *Cortex: Statistics and geometry of neuronal connectivity*. Springer-Verlag Berlin Heidelberg, 1998.

[5] A. Cangelosi. Grounding language in action and perception: From cognitive agents to humanoid robots. *Physics of Life Reviews*, 7(2):139–151, 2010.

[6] A. Cangelosi and M. Schlesinger. *Developmental robotics: From babies to robots*. The MIT Press, 2015.

[7] R. M. Cichy, A. Khosla, D. Pantazis, T. Antonio, and A. Oliva. Comparison of deep neural networks to spatio-temporal cortical dynamics of human visual object recognition reveals hierarchical correspondence. *Scientific Reports*, 6:13 p., 2016.

[8] S. Heinrich and S. Wermter. Interactive language understanding with multiple timescale recurrent neural networks. In *Proc. 24th ICANN*, volume 8681 of *LNCS*, pages 193–200, 2014.

[9] S. Heinrich and S. Wermter. Interactive natural language acquisition in a multi-modal recurrent neural architecture. *Connection Science*, under review, 2016/2017.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS 2012*, pages 1097–1105, 2012.

[11] W. J. M. Levelt. Spoken word production: A theory of lexical access. *Proceedings of the National Academy of Sciences of the United States of America*, 98(23):13464–13471, 2001.

[12] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proc. AAAI-15*, pages 2181–2187, 2015.

[13] F. Pulvermüller, M. Garagnani, and T. Wennekers. Thinking in circuits: toward neurobiological explanation in cognitive neuroscience. *Biological Cybernetics*, 108(5):573–593, 2014.

[14] D. K. Roy. Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1–2):170–205, 2005.

[15] L. Steels and M. Hild. *Language Grounding in Robots*. Springer Science+Business Media LLC New York, 2012.

[16] J. Tani. Self-organization and compositionality in cognitive brains: A neurorobotics study. *Proceedings of the IEEE*, 102(4):586–605, 2014.

[17] M. Vavrečka and I. Farkaš. A multimodal connectionist architecture for unsupervised grounding of spatial language. *Cognitive Computation*, 6(1):101–112, 2014.

[18] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun. Representation learning of knowledge graphs with entity descriptions. In *Proc. AAAI-16*, pages 2659–2665, 2016.

[19] Y. Yamashita and J. Tani. Emergence of functional hierarchy in a multiple timescale neural network model: A humanoid robot experiment. *PLOS Computational Biology*, 4(11):e1000220, 2008.

Diagnostic classifiers: revealing how neural networks process hierarchical structure

Sara Veldhoen, Dieuwke Hupkes and Willem Zuidema
ILLC, University of Amsterdam
P.O.Box 94242, 1090 CE Amsterdam, Netherlands
{s.f.veldhoen, d.hupkes, zuidema}@uva.nl

Abstract

We investigate how neural networks can be used for hierarchical, compositional semantics. To this end, we define the simple but nontrivial artificial task of processing nested arithmetic expressions and study whether different types of neural networks can learn to add and subtract. We find that recursive neural networks can implement a generalising solution, and we visualise the intermediate steps: projection, summation and squashing. We also show that gated recurrent neural networks, which process the expressions incrementally, perform surprisingly well on this task: they learn to predict the outcome of the arithmetic expressions with reasonable accuracy, although performance deteriorates with increasing length. To analyse what strategy the recurrent network applies, visualisation techniques are less insightful. Therefore, we develop an approach where we formulate and test hypotheses on what strategies these networks might be following. For each hypothesis, we derive predictions about features of the hidden state representations at each time step, and train 'diagnostic classifiers' to test those predictions. Our results indicate the networks follow a strategy similar to our hypothesised 'incremental strategy'.

1 Introduction

A key property of language is its hierarchical compositional semantics: the meaning of larger wholes such as phrases and sentences depends on the meaning of the words they are composed of and the way they are put together, and phrases can be hierarchically combined into more complex phrases. For example, the meaning of the compound noun *natural language research paper* is a combination of the meanings of *natural language* and *research paper*, which in turn are combinations of the meanings of the individual words. Whether 'classical' neural networks can adequately compute the grammaticality and meaning of sentences with hierarchical structure has been a the topic of many heated debates in linguistics and cognitive science [e.g. Pinker and Mehler, 1988, Fodor and Pylyshyn, 1988].

In the literature, one can find many attempts at designing neural models that are able to capture hierarchical compositionality [e.g. Elman, 1990, Rodriguez, 2001, Sutskever et al., 2011], as well as a vast amount of papers on improved computational methods to train such models [e.g. Zeiler, 2012, Kingma and Ba, 2014], but very few focus on understanding the internal dynamics of the parameter-rich models [but see Karpathy et al., 2015]. In this paper, we do not focus on designing better training algorithms or models, or even on achieving high accuracies, but rather on understanding *how* neural networks can implement solutions that involve hierarchical compositionality.

A substantial problem encountered in such an undertaking, is the difficulty of evaluating whether a neural model can in fact capture the semantics of a sentence, as meanings can usually not be summarised by a single number or class. Several tools have been developed to assess the quality of

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

```
result_stack = [], mode_stack = [];
result = 0, mode = +;
for symbol in expression do
    if symbol == '(' then
        mode_stack.append(mode);
        result_stack.append(result);
        result = 0
    else if symbol == ')' then
        mode = mode_stack.pop();
        prev_result = result_stack.pop();
        result = mode(prev_result, result)
    else if symbol == '+' then
        mode = +
    else if symbol == '-' then
        mode = -
    else
        result = apply(mode, result, symbol)
end
return result
```

(a) Recursive strategy

```
mode_stack = [];
result = 0, mode = +;
for symbol in expression do
    if symbol == '(' then
        mode_stack.append(mode)
    else if symbol == ')' then
        mode = mode_stack.pop()
    else if symbol == '+' then
        pass
    else if symbol == '-' then
        if mode == - then
            mode = +
        else
            mode = -
    else
        result = apply(mode, result, symbol)
end
return result
```

(b) Incremental strategy

Figure 1: Different symbolic strategies for incrementally solving arithmetic expressions incrementally. The `apply` function applies the operator specified by *mode* $(+, -)$ to the two numbers specified by *result* and *symbol*.

lexical representations, but no satisfactory option is available to study representations that result from composition. A common approach to train and evaluate sentence representations is to have them execute a task that requires semantic knowledge of the sentence such as sentiment analysis, logical inference or question-answering [e.g. Le and Zuidema, 2015a,b, Hermann et al., 2015].

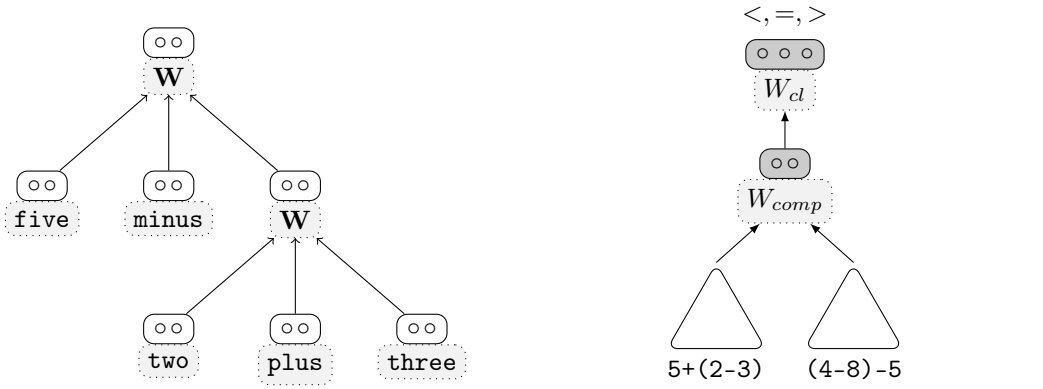
Although solving such tasks might be worth pursuing in its own right, this task-driven setting obscures the mechanisms for semantic composition that we aim to investigate. Therefore, in this paper we take a different approach, by looking at an artificial language of which both the sentences and the lexical items have clearly (and symbolically) defined meanings. We present a thorough analysis of two different kinds of neural models that process sentences from this language, as well as an innovative method to study their behaviour: the diagnostic classifier.

2 Arithmetic language

The language we consider for this study consists of words for all integers in the range $\{-10, \ldots, 10\}$, the operators `+` and `-` and the brackets `(` and `)`. All (bracketed) mathematically correct expressions that can be formed with these words constitute the sentences of the arithmetic language. The (compositional) meaning of a sentence is the solution of the corresponding arithmetic expression. As the minus operator is non-commutative, meanings depend not only on lexical items, but also on the syntactic structure as prescribed by the brackets. The unambiguously defined syntax and semantics, as well as their entanglement, make the arithmetic language very suitable to evaluate whether and how neural network models can learn compositional semantics.

Naturally, we do not aim to find or improve ways to solve the task of computing the outcome of arithmetic expressions. Rather, we study the solutions found by neural networks, using our understanding of how the task can be solved symbolically. In what follows, we will distinguish sentences by length, that is: 'L7' refers to those sentences that contain exactly 7 digits. We also distinguish subsets containing only left-branching (e.g. L9L) or right-branching expressions (L6R), respectively, as well as syntactically non-restricted subsets.

Symbolic strategies One can think of different strategies to incrementally compute the solution of an arithmetic expression, the most obvious of which perhaps involves recursively traversing through the expression and computing the outcome of all subtrees before finally accumulating the results at the end. Alternatively, the digits can be accumulated immediately at the moment they are encountered, maintaining a prediction of the solution at any point during the computation. Figure 1



(a) A TreeRNN to represent `( 5 + ( 2 - 3 ) )`

(b) Training setup: compare two expressions from concatenation of root representations

Figure 2: Network architecture TreeRNN

contains procedural descriptions for both the recursive and the incremental strategy. An incremental implementation of the recursive strategy requires a stack to store previously computed outcomes of subtrees, as well as the operators needed to integrate them. The incremental strategy, on the other hand, requires only information about previously seen operators, which is less demanding.

3 Models

We study two types of network architectures: recursive neural networks and recurrent neural networks. The former are structured following the syntactic structure of a sentence, whereas the latter processes sentences sequentially, reading them one word at the time.

3.1 Recursive Neural Network (TreeRNN)

The Recursive Neural Network or *TreeRNN* [Socher et al., 2010, Goller and Kuchler, 1996] is a hybrid neural network model that explicitly deals with compositionality by shaping its architecture after the syntactic structure of its input. This structure is thus assumed to be known beforehand, when the network is instantiated, which makes the network dependent on external parses. Figure 2a shows a TreeRNN instantiation of an L3 expression.

The composition function of the TreeRNN

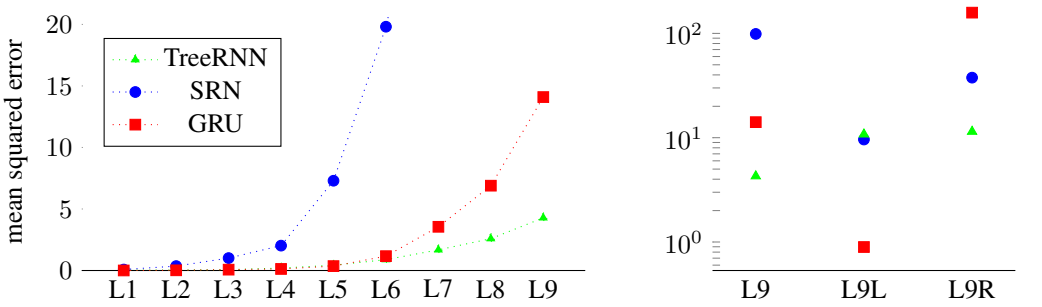
$$\mathbf{p} = \tanh(\mathbf{W} \cdot [\mathbf{x}_1; \mathbf{x}_2; \mathbf{x}_3] + \mathbf{b}), \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{d \times 3d}$ and $\mathbf{b} \in \mathbb{R}^d$, operates on the concatenation of three d-dimensional vectors $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}$ that represent subphrases or words. The result is a vectorial parent representation $\mathbf{p}$ of the same dimensionality d. As such, the function can be applied recursively. Notably, there is a single composition function that is applied for each subtree, where the operators are treated as lexical items.

Training The composition function is trained in a regime inspired by Bowman and Potts [2015], who train a network for logical reasoning. Two sentence meanings are computed by treeRNNs with shared parameters. An additional neural classifier predicts the relation between the concatenated sentence representations, which generates an error signal that is backpropagated to update the parameters. In this case, the prediction concerns the (in)equality ('<', '=' or '>') that holds between the two (numerical) solutions. This training setup is displayed in Figure 2b.

3.2 Recurrent Neural Network (RNN)

In addition to the syntactically informed TreeRNNs, we study the behaviour of their well-known sequential counterparts: recurrent neural networks (RNNs). In particular, we consider two types of RNNs: simple recurrent networks (SRNs), as originally introduced by Elman [1990], and Gated



(a) Performance (MSE) with respect to sentence length

(b) MSE (log scale) on L9 and L9L (only left-branching) and L9R (only right-branching)

Figure 3: Mean Squared Error of scalar prediction for best performing models on the test sentences.

Recurrent Units (GRUs), defined in [Cho et al., 2014]. The SRN consist of a single hidden layer $\mathbf{h}$ with a nonlinear activation function and a feedback connection:

$$\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{\mathbf{t-1}} + \mathbf{b}) \tag{2}$$

The GRU is an extended version of this model, in which *gates* are used to modulate the recurrent information flow. Two gate values are computed from the previous hidden layer activation and the current input. The reset gate $\mathbf{r}$ affects the computation of a candidate hidden state $\tilde{\mathbf{h}}_t$:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r) \qquad \tilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}_h(\mathbf{r} \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{3}$$

where $\odot$ denotes the element wise product and σ the logistic sigmoid function. The new activation $\mathbf{h}_{t-1}$ is a weighted sum of the candidate activation $\tilde{\mathbf{h}}_t$ and the previous activation $\mathbf{h}_{t-1}$, modulated by the update gate $\mathbf{z}$.

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z) \qquad \mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t, \tag{4}$$

All $\mathbf{W} \in \mathbb{R}^{h \times d}$, all $\mathbf{U} \in \mathbb{R}^{h \times h}$ and all $\mathbf{b}$, $\mathbf{z}$ and $\mathbf{r} \in \mathbb{R}^h$ with $h = |\mathbf{h}|$ and $d = |\mathbf{x}|$.

Training The RNNs have access to the syntactic structure implicitly, through the brackets, inputted as words. Both the embeddings and the weights for the sequential models are trained on an error signal from a simple linear perceptron that predicts the real-valued solution of the expression from the hidden representation in the last time step.

4 Data and Results

We train all models on an arbitrarly sampled set of expressions from L1, L2, L4, L5 and L7 (3000 expressions from each subset) with randomly generated syntactic structures. During training we update also the word embeddings. All models are trained using stochastic gradient descent with minibatches. The scalar prediction models are trained with optimiser Adam [Kingma and Ba, 2014] and `mse` loss, the original TreeRNNs with Adagrad [Zeiler, 2012] and cross-entropy.[1]

The TreeRNN obtains a classification accuracy of 0.97 on test data (sentences up to length 9) for the comparison task. To make its results comparable to those of the sequential RNNs, we train a neural network with one hidden layer to perform the same task as the RNNs, i.e., predicting the solution of expressions. Importantly, we keep both the embeddings and the parameters of the TreeRNN fixed during this training, and only update the weights of the prediction network.

The main purpose of quantifying how well the task is performed by different types networks is to establish which models learned to perform the task well enough to analyse *how* they implement the

[1]Source code of the simulations can be found at `https://github.com/dieuwkehupkes/Arithmetics`. For part of our implementation we used the Python library Keras [Chollet, 2015].

solution. We therefore only report the results of the models that perform best, which can be found in Figure 3.[2] The SRNs do not seem to have learned a principled solution with this training regime. We find that the TreeRNN generalises adequately to longer (unseen) sentences, evident from the smooth progression of the performance with respect to length [Bowman et al., 2015]. Although the generalisation capacity of the GRU (with $|h| = 15$) is weaker, it seems the GRU has also learned a solution that is sensitive to the syntactic structure. Figure 3b reveals that completely right-branching sentences are much harder to process for the GRU. Left-branching sentences, which allow for direct accumulation of numbers, prove much easier. The performance of TreeRNNs, on the other hand, suffers from tree depth, regardless of the branching direction.

5 Analysis

In the previous section, we established that a number of our trained networks can - at least to some extent - compute the compositional meaning of sentences and phrases in the arithmetic language in a generalising way. We will present an analysis of the internal dynamics of the best performing models, explaining *how* the task of computing the outcome of the arithmetic expressions is implemented. We will start with considering the TreeRNN, whose extremely low dimensionality allows for a comprehensible account of the implementation of the solution. Subsequently, we will analyse the internal dynamics of the GRU network, and introduce a new approach for testing hypotheses about the strategies a network might engage.

5.1 TreeRNN

To understand how a TreeRNN computes the composition of digits and operators, we break down the computation into different steps: project, sum and squash.

Project The application function of the TreeRNN (1) can be rewritten as a sum of three terms, which results in the following expression for the composition function:

$$\mathbf{p} = f(\mathbf{W}_1 \cdot \mathbf{x}_L + \mathbf{W}_M \cdot \mathbf{x}_2 + \mathbf{W}_R \cdot \mathbf{x}_3 + \mathbf{b}) \tag{5}$$

where each $\mathbf{W}_i \in \mathbb{R}^{2\times 2}$ projects one of the three input children to a new position in the two dimensional representational space. Figure 4 depicts the digit embeddings and their projections, illustrating how the left and right children are projected to almost orthogonal subspaces.

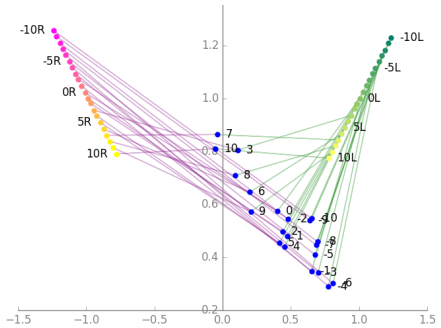
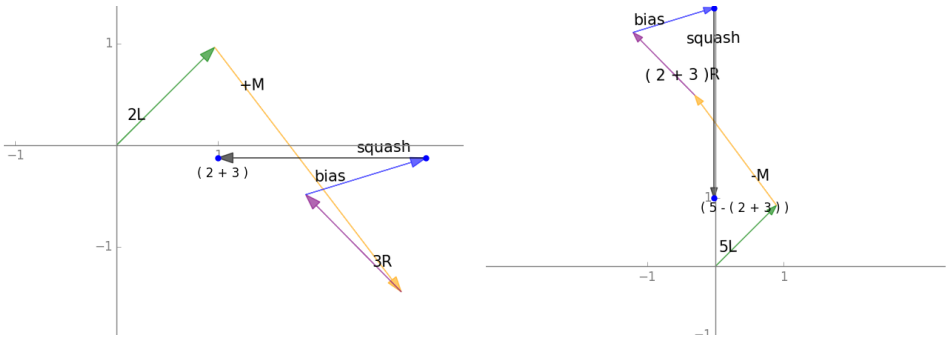


Figure 4: The projections of the seemingly unorganised digit embeddings (blue) through $\mathbf{W}_L$ (as left child, green) and $\mathbf{W}_R$ (as right child, purple) reveal their organisation.

Sum The coloured arrows in Figure 5 show how the projected representations and the bias term are summed. From the two subfigures, it is clear that the projections of the operators + and - through $\mathbf{W}_M$ are roughly opposite. Together with the bias, the result is sent to positions close to the y- and x-axis, for + and - respectively, which is important for the last step of the computation.

[2]Except for the SRN models, the results over different runs were comparable on a performance level.

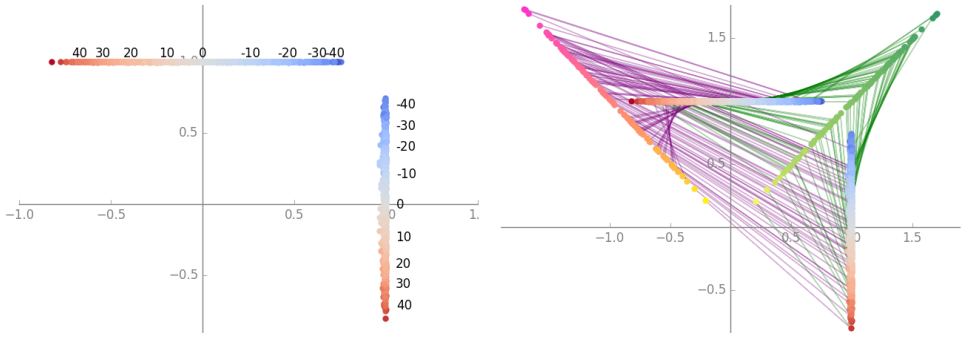


(a) (2 + 3) (b) (5 - '(2 + 3)')

Figure 5: Summation of projections - left (green), middle (yellow) and right (purple) child - together with bias term (blue). Squash ($\tanh$ activation, grey) shifts the result back to the range $(-1, 1)$.

Squash In this last step, both coordinates of the representation are squashed into the range $(-1, 1)$ by the activation function `tanh`. As the projected representations are already within this range for one of the dimensions, depending on the operator, the effect of the squash operation is close to a horizontal or vertical shift.

Recursion Figure 6a depicts the representations of 2000 L9 expressions whose last applied operators (under the recursion) were subtraction (representations on horizontal line) and addition (vertical line), respectively. In Figure 6b we see that if these representations are the input of a new operation, they get projected onto the same diagonals (by the left and right part of the composition matrix, see equation 5) as the digit embeddings were. Although noise may be accumulated through the depth of the tree, this solution thus works recursively for longer expressions.



(a) Representations of L9 sentences (b) Projections of L9 representations

Figure 6: Representations of 2000 L9 expressions whose root operation was subtraction ($x \approx 1$, vertical line) or addition ($y \approx 1$, horizontal line). Right: how these would get projected by $\mathbf{W}_L$ (green) and $\mathbf{W}_R$ (purple) to serve as an input to a new computation.

5.2 Sequential model

Studying the internal structure of the sequential RNNs is more complicated, as the higher dimensionality of the models makes visual inspection impractical. Furthermore, the complex interaction between gates and hidden states in the GRU, as well as the fact that the networks are forced to incrementally process an input that is not strictly locally processable, makes it much less clear which computation is carried out at what point in time.

Diagnostic Classifiers To analyse the internal dynamics of the GRU, we propose a generic method that can be used to test and formulate hypotheses about solutions a model could be implementing on an algorithmic level [Marr, 1982]. Such hypotheses are formulated as sequences of targets for each

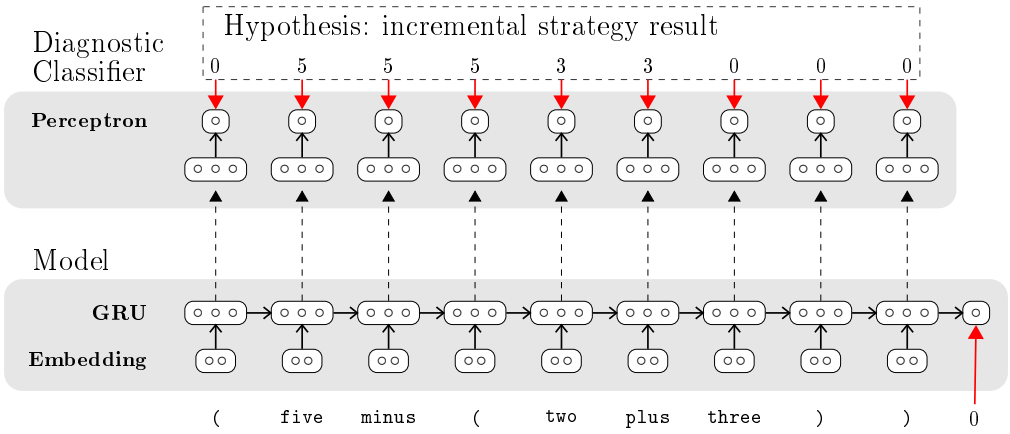
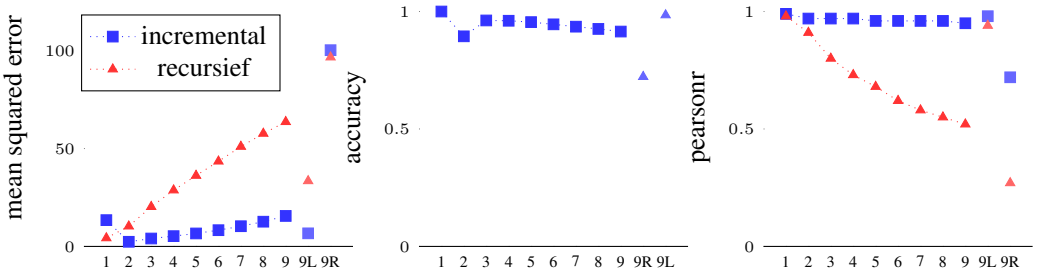


Figure 7: Testing a single hypothesis with a diagnostic classifier.

time step, and *diagnostic classifiers* are trained to predict such sequences from the RNNs hidden representations. Evidently, the parameters of the original model are kept fixed during training.

For example, if the network were to follow the incremental strategy described in Figure 1b, the model should have a representation of the intermediate `result` at each point in time. If the sequence of intermediate results can be accurately predicted by a diagnostic classifier (see Figure 7), this indicates that these values are in fact computed by the original model. As the diagnostic model should merely read out whether certain information is present in the hidden representations rather than perform complex computations itself, we use a simple linear model as diagnostic classifier. In the current paper, we test the hypothesis that the network follows either the incremental or the recursive strategy described in Figure 1. To this end, we train three diagnostic classifiers to predict the values of the intermediate results (`result` in Figure 1) and the variable `mode` used by the intermediate strategy to determine whether the next number should be added or subtracted.

Results We find that the values required for the incremental strategy (`mode` and `result`) can be more accurately predicted than the recursive intermediate strategy values (see Figure 8). From these findings it appears unlikely that the network implements a fully recursive strategy employing a stack of intermediate results. For the incremental strategy, on the other hand, the predictions are generally accurate, even for longer sentences. Notably, the diagnostic models exhibit a substantially lower accuracy for right-branching trees than for lengthy trees that are entirely left-branching. This is consistent with assumptions about the difficulty of employing information on larger size stacks as, contrary to right-branching trees, left-branching trees can be processed strictly locally. However, relatively high errors for sentences from L1 and L2 (for `result` and `mode` prediction, respectively) reveal that the network deviates from this strategy at least at some points.



(a) prediction of `result` (b) `mode` incremental (c) trajectory correlations

Figure 8: Results of diagnostic models for GRU on different subsets of languages

Trajectories A similar conclusion can be drawn from studying the predictions of the diagnostic classifiers. The predictions of the diagnostic classifiers on two randomly picked L9 sentences, along with their target trajectories as defined by the hypotheses, are depicted in Figure 9. These trajectories confirm that the line representing the incremental strategy is much better tracked than the recursive one (a correlation test over 5000 L9 sentences shows the same trend: Pearson's r = 0.52 and 0.95 for recursive and incremental, respectively).[3]

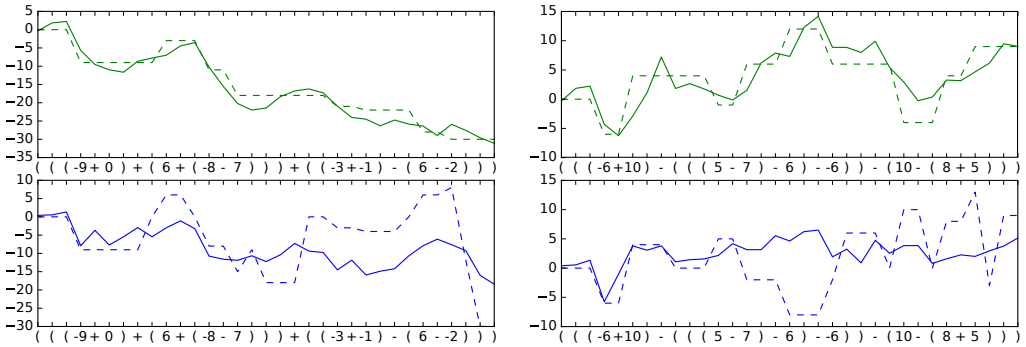


Figure 9: Trajectories of the incremental (green, upper) and recursive (blue, lower) classifier, along with their targets. Dashed lines show target trajectories.

6 Conclusions

In this paper we studied how recursive and recurrent neural networks process hierarchical structure, using a simple arithmetic language as a convenient toy task with unambiguous syntax and semantics and a limited vocabulary. We showed that recursive neural networks can learn to compute the meaning of arithmetic expressions and readily generalise to expressions longer than any seen in training. Learning was even successful when the representations were limited to two dimensions allowing a geometrical analysis of the solution found by the network. Understanding the organisation of the word embedding space and the nature of the projections encoded in the learned composition function, combined with vector addition and squashing, gave us a complete picture of the compositional semantics that the network has learned. This made clear that the network has found a near perfect approximation of a principled, recursive solution of the arithmetic semantics.

Still, the recursive neural network is a hybrid symbolic connectionist architecture, as the network architecture directly reflects the tree structure of the expression it must process, and this architecture is built up anew for each new expression using a symbolic control structure. This limits both the computational efficiency and the usefulness of this model for understanding how the human brain might process hierarchical structure. In this paper we therefore also analyse two recurrent neural network architectures: the classical Simple Recurrent Network and the recently popular Gated Recurrent Units. As it turns out, the latter can also learn to compute the meaning of arithmetic expressions and generalise to longer expressions than seen in training. Understanding how this network solves the task, however, is more difficult due to its higher dimensionality, recurrent connections and gating mechanism. To better understand what the network is doing we therefore developed an approach based on training diagnostic classifiers on the internal representations.

The qualitative and quantitative analysis of the results of a diagnostic classifier allows us to draw conclusions about possible strategies the network might be following. In particular, we find that the successful networks follow a strategy very similar to our hypothesised symbolic 'incremental strategy'. From this we learn something about how neural networks may process languages with a hierarchical compositional semantics and, perhaps more importantly, also provides an example of how we can 'open the black box' of the many successful deep learning models in natural language processing (and other domains), when visualisation alone is not sufficient.

[3]The plots in Figure 9 also point to possible further refinements of the hypothesis: where the predictions of the model change at every point in time, the targets value often remains the same for longer time spans, indicating that a hypothesis in which information is accumulated more gradually would give an even better.

References

Samuel R Bowman and Christopher Potts. Recursive neural networks can learn logical semantics. *ACL-IJCNLP 2015*, page 12, 2015.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. Tree-structured composition in neural networks without tree-structured architectures. *CoRR*, abs/1506.04834, 2015. URL `http://arxiv.org/abs/1506.04834`.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Christoph Goller and Andreas Kuchler. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE, 1996.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Phong Le and Willem Zuidema. Compositional distributional semantics with long short term memory. *arXiv preprint arXiv:1503.02510*, 2015a.

Phong Le and Willem Zuidema. The forest convolutional network: Compositional distributional semantics with a neural chart and without binarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1155–1164, 2015b.

DC Marr. A computational investigation into the human representation and processing of visual information, 1982.

Steven Pinker and Jacques Mehler. *Connections and symbols*, volume 28. Mit Press, 1988.

Paul Rodriguez. Simple recurrent networks learn context-free and context-sensitive languages by counting. *Neural Computation*, 13(9):2093–2118, 2001.

Richard Socher, Christopher D Manning, and Andrew Y Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9, 2010.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Neuro-symbolic EDA-based Optimisation using ILP-enhanced DBNs

Sarmimala Saikia
TCS Research, New Delhi
sarmimala.saikia@tcs.com

Lovekesh Vig
TCS Research, New Delhi
lovekesh.vig@tcs.com

Ashwin Srinivasan
Department of Computer Science
BITS Goa
ashwin@goa.bits-pilani.ac.in

Gautam Shroff
TCS Research, New Delhi
gautam.shroff@tcs.com

Puneet Agarwal
TCS Research, New Delhi
puneet.a@tcs.com

Richa Rawat
TCS Research, New Delhi
rawat.richa@tcs.com

Abstract

We investigate solving discrete optimisation problems using the 'estimation of distribution' (EDA) approach via a novel combination of deep belief networks (DBN) and inductive logic programming (ILP). While DBNs are used to learn the structure of successively 'better' feasible solutions, ILP enables the incorporation of domain-based background knowledge related to the goodness of solutions. Recent work showed that ILP could be an effective way to use domain knowledge in an EDA scenario. However, in a purely ILP-based EDA, sampling successive populations is either inefficient or not straightforward. In our Neuro-symbolic EDA, an ILP engine is used to construct a model for good solutions using domain-based background knowledge. These rules are introduced as Boolean features in the last hidden layer of DBNs used for EDA-based optimization. This incorporation of logical ILP features requires some changes while training and sampling from DBNs: (a) our DBNs need to be trained with data for units at the input layer as well as some units in an otherwise hidden layer; and (b) we would like the samples generated to be drawn from instances entailed by the logical model. We demonstrate the viability of our approach on instances of two optimisation problems: predicting optimal depth-of-win for the KRK endgame, and job-shop scheduling. Our results are promising: (i) On each iteration of distribution estimation, samples obtained with an ILP-assisted DBN have a substantially greater proportion of good solutions than samples generated using a DBN without ILP features; and (ii) On termination of distribution estimation, samples obtained using an ILP-assisted DBN contain more near-optimal samples than samples from a DBN without ILP features. Taken together, these results suggest that the use of ILP-constructed theories could be useful for incorporating complex domain-knowledge into deep models for estimation of distribution based procedures.

1 Introduction

There are many real-world planning problems for which domain knowledge is qualitative, and not easily encoded in a form suitable for numerical optimisation. Here, for instance, are some guiding principles that are followed by the Australian Rail Track Corporation when scheduling trains: (1) If a healthy Train is running late, it should be given equal preference to other healthy Trains; (2) A higher priority train should be given preference to a lower priority train, provided the delay to the lower priority train is kept to a minimum; and so on. It is evident from this that train-scheduling may benefit from knowing if a train is healthy, what a trains priority is, and so on. But are priorities

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

and train-health fixed, irrespective of the context? What values constitute acceptable delays to a low-priority train? Generating good train-schedules will require a combination of quantitative knowledge of train running times and qualitative knowledge about the train in isolation, and in relation to other trains. In this paper, we propose a heuristic search method, that comes under the broad category of an estimation distribution algorithm (EDA). EDAs iteratively generate better solutions for the optimisation problem using machine-constructed models. Usually EDAs have used generative probabilistic models, such as Bayesian Networks, where domain-knowledge needs to be translated into prior distributions and/or network topology. In this paper, we are concerned with problems for which such a translation is not evident. Our interest in ILP is that it presents perhaps one of the most flexible ways to use domain-knowledge when constructing models. Recent work has shown that ILP models incorporating background knowledge were able to generate better quality solutions in each EDA iteration [14]. However, efficient sampling is not straightforward and ILP is unable to utilize the discovery of high level features as efficiently as deep generative models.

While neural models have been used for optimization [15], in this paper we attempt to combine the sampling and feature discovery power of deep generative models with the background knowledge captured by ILP for optimization problems that require domain knowledge. The rule based features discovered by the ILP engine are appended to the higher layers of a Deep Belief Network(DBN) while training. A subset of the features are then clamped on while sampling to generate samples consistent with the rules. This results in consistently improved sampling which has a cascading positive effect on successive iterations of EDA based optimization procedure. The rest of the paper is organised as follows. Section 2 provides a brief description of the EDA method we use for optimisation problems. Section 2.1 describes how ILP can be used within the iterative loop of an EDA for discovering rules that would distinguish good samples from bad. Section 3 Describes how RBMs can be used to generate samples that conform to the rules discovered by the ILP engine. Section 4 describes an empirical evaluation demonstrating the improvement in the discovery of optimal solutions, followed by conclusions in Section 5.

2 EDA for optimization

The basic EDA approach we use is the one proposed by the MIMIC algorithm [4]. Assuming that we are looking to minimise an objective function $F(\mathbf{x})$, where $\mathbf{x}$ is an instance from some instance-space $\mathcal{X}$, the approach first constructs an appropriate machine-learning model to discriminate between samples of lower and higher value, i.e., $F(\mathbf{x}) \leq \theta$ and $F(\mathbf{x}) > \theta$, and then generates samples using this model

Procedure EODS: Evolutionary Optimisation using DBNs for Sampling

1. Initialize population $P := \{\mathbf{x}_i\}$; $\theta := \theta_0$
2. while not converged do
   (a) for all $\mathbf{x}_i$ in P $label(\mathbf{x}_i)$:= 1 if $F(\mathbf{x}_i) \leq \theta$ else $label(\mathbf{x}_i)$:= 0
   (b) train DBN M to discriminate between 1 and 0 labels i.e., $P(\mathbf{x} : label(\mathbf{x}) = 1|M) > P(\mathbf{x} : label(\mathbf{x}) = 0|M)$
   (c) regenerate P by repeated sampling using model M
   (d) reduce threshold θ
3. return P

Figure 1: Evolutionary optimisation using a network model to generate samples.

Here we use Deep Belief Networks (DBNs) [7] for modeling our data distribution, and for generating samples for each iteration of MIMIC. Deep Belief Nets (DBNs) are generative models that are composed of multiple latent variable models called Restricted Boltzman Machines (RBMs).In particular, as part of our larger optimization algorithm, we wish to repeatedly train and then sample from the trained DBN in order to reinitialize our sample population for the next iteration as outlined in Figure 1. In order to accomplish this, while training we append a single binary unit (variable) to the highest hidden layer of the DBN, and assign it a value 1 when the value of the sample is below θ and a value 0 if the value is above θ. During training that this variable, which we refer to as the separator variable, learns to discriminate between good and bad samples. To sample from the DBN we additionally clamp our separator variable to 1 so as to bias the network to produce good samples,

and preserve the DBN weights from the previous MIMIC iteration to be used as the initial weights for the subsequent iteration. This prevents retraining on the same data repeatedly as the training data for one iteration subsumes the samples from the previous iteration.

We now look at how ILP models can assist DBNs constructed for this purpose.

3 EDA using ILP-assisted DBNs

3.1 ILP

The field of Inductive Logic Programming (ILP) has made steady progress over the past two and half decades, in advancing the theory, implementation and application of logic-based relational learning. A characteristic of this form of machine-learning is that data, domain knowledge and models are usually—but not always—expressed in a subset of first-order logic, namely logic programs. Side-stepping for the moment the question "why logic programs?", domain knowledge (called *background knowledge* in the ILP literature) can be encodings of heuristics, rules-of-thumb, constraints, text-book knowledge and so on. It is evident that the use of some variant of first-order logic enable the automatic construction of models that use relations (used here in the formal sense of a truth value assignment to n-tuples). Our interest here is in a form of relational learning concerned with the identification of functions (again used formally, in the sense of being a uniquely defined relation) whose domain is the set of instances in the data. An example is the construction of new *features* for data analysis based on existing relations ("$f(m) = 1$ if a molecule m has 3 or more benzene rings fused together otherwise $f(m) = 0$": here concepts like benzene rings and connectivity of rings are generic relations provided in background knowledge).

There is now a growing body of research that suggests that ILP-constructed relational features can substantially improve the predictive power of a statistical model (see, for example: [9, 11, 12, 10, 13]). Most of this work has concerned itself with discriminatory models, although there have been cases where they have been incorporated within generative models. In this paper, we are interested in their use within a deep network model used for generating samples in an EDA for optimisation in Procedure EODS in Fig. 1.

3.2 ILP-assisted DBNs

Given some data instances $\mathbf{x}$ drawn from a set of instances $\mathcal{X}$ and domain-specific background knowledge, let us assume the ILP engine will be used to construct a model for discriminating between two classes (for simplicity, called $good$ and bad). The ILP engine constructs a model for $good$ instances using rules of the form $h_j : Class(\mathbf{x}, good) \leftarrow Cp_j(\mathbf{x})$.[1] $Cp_j : \mathcal{X} \mapsto \{0, 1\}$ denotes a "context predicate". A context predicate corresponds to a conjunction of literals that evaluates to $TRUE$ (1) or $FALSE$ (0) for any element of $\mathcal{X}$. For meaningful features we will usually require that a Cp_j contain at least one literal; in logical terms, we therefore require the corresponding h_j to be definite clauses with at least two literals. A rule $h_j : Class(\mathbf{x}, good) \leftarrow Cp_j(\mathbf{x})$, is converted to a feature f_j using a one-to-one mapping as follows: $f_j(\mathbf{x}) = 1$ iff $Cp_j(\mathbf{x}) = 1$ (and 0 otherwise). We will denote this function as $Feature$. Thus $Feature(h_j) = f_j$, $Feature^{-1}(f_j) = h_j$. We will also sometimes refer to $Features(H) = \{f : h \in H \text{ and } f = Feature(h)\}$ and $Rules(F) = \{h : f \in F \text{ and } h = Features^{-1}(f)\}$.

Each rule in an ILP model is thus converted to a single Boolean feature, and the model will result in a set of Boolean features. Turning now to the EODS procedure in Fig. 1, we will construct ILP models for discriminating between $F(\mathbf{x}) \leq \theta$) ($good$) and $F(\mathbf{x}) > \theta$ (bad). Conceptually, we treat the ILP-features as high-level features for a deep belief network, and we append the data layer of the highest level RBM with the values of the ILP-features for each sample as shown in Fig 2.

[1] We note that in general for ILP $\mathbf{x}$ need not be restricted to a single object and can consist of arbitrary tuples of objects and rules constructed by the ILP engine would more generally be $h_j : Class(\langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \rangle, c) \leftarrow Cp_j(\langle \mathbf{x_1}, \mathbf{x}_2, \ldots, \mathbf{x}_n \rangle)$. But we do not require rules of this kind here.

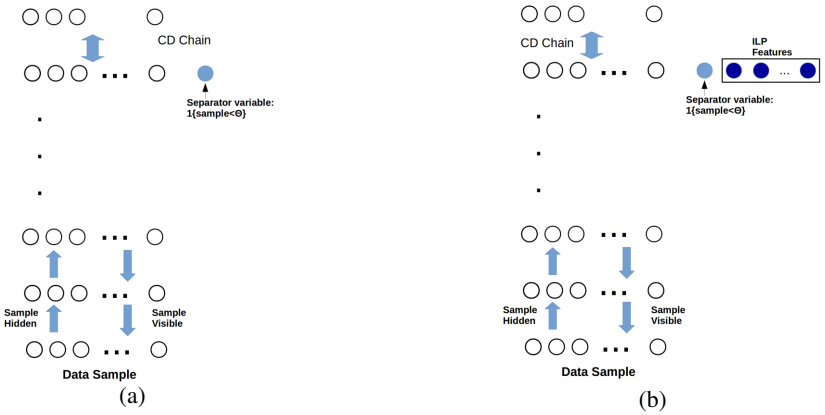


Figure 2: Sampling from a DBN (a) with just a separator variable (b) with ILP features

3.3 Sampling from the Logical Model

Recent work [14] suggests that if samples can be drawn from the success-set of the ILP-constructed model[2] then the efficiency of identifying near-optimal solutions could be significantly enhanced. A straightforward approach of achieving this with an ILP-assisted DBN would appear to be to clamp all the ILP-features, since this would bias the the samples from the network to sample from the intersection of the success-sets of the corresponding rules (it is evident that instances in the intersection are guaranteed to be in the success-set sought). However this will end up being unduly restrictive, since samples sought are not ones that satisfy all rules, but *at least* one rule in the model. The obvious modification would be to clamp subsets of features. But not all samples from a subset of features may be appropriate.

With a subset of features clamped, there is an additional complication that arises due to the stochastic nature of the DBN's hidden units. This makes it possible for the DBN's unit corresponding to a logical feature f_j to have the value 1 for an instance $\mathbf{x}_i$, but for $\mathbf{x_i}$ not to be entailed by the background knowledge and logical rule h_j.

In turn, this means that for a feature-subset with values clamped, samples may be generated from outside the success-set of corresponding rules involved. Given background knowledge B, we say a sample instance $\mathbf{x}$ generated by clamping a set of features F is *aligned* to $H = Rules(F)$, iff $B \wedge H \models \mathbf{x}$ (that is, $\mathbf{x}$ is entailed by B and H).

A procedure to bias sampling of instances from the success-set of the logical model constructed by ILP is shown in Fig. 3.

4 Empirical Evaluation

4.1 Aims

Our aims in the empirical evaluation are to investigate the following conjectures:

(1) On each iteration, the EDOS procedure will yield better samples with ILP features than without

(2) On termination, the EDOS procedure will yield more near-optimal instances with ILP features than without.

(3) Both procedures do better than random sampling from the initial training set.

It is relevant here to clarify what the comparisons are intended in the statements above. Conjecture (1) is essentially a statement about the gain in precision obtained by using ILP features. Let us denote

[2]These are the instances entailed by the model along with the background knowledge, which—assuming the rules are not recursive—we take to be the union of the success-sets of the individual rules in the model.

Given: Background knowledge B; a set of rules $H = \{h_1, h_2, \ldots, h_N\}$l; a DBN with $F = \{f_1, f_2, \ldots, f_N\}$ as high-level features ($f_i = Feature(h_i)$); and a sample size M

Return: A set of samples $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ drawn from the success-set of $B \wedge H$.

1. $S := \emptyset, k = 0$
2. while $|k| \leq N$ do
   (a) Randomly select a subset F_k of size k features from F
   (b) Generate a small sample set X clamping features in F_k
   (c) for each sample in $x \in X$ and each rule h_j, set $count_k = 0$
      i. if $x \in$ success-set where $(f_j(x) = 1) => (x \in$ success-set(B and h_j)) $count_k = count_k + 1$
3. Generate S by clamping k features where $count_k = max(count_1, count_2 ... count_N)$
4. return S

Figure 3: A procedure to generate samples aligned to a logical model H constructed by an ILP engine.

$Pr(F(\mathbf{x}) \leq \theta)$ the probability of generating an instance $\mathbf{x}$ with cost at most θ without ILP features to guide sampling, and by $Pr(F(\mathbf{x}) \leq \theta | M_{k,B})$ the probability of obtaining such an instance with ILP features $M_{k,B}$ obtained on iteration k of the EDOS procedure using some domain-knowledge B. (note if $M_{k,B} = \emptyset$, then we will mean $Pr(F(\mathbf{x}) \leq \theta | M_{k,B}) = Pr(F(\mathbf{x}) \leq \theta)$). Then for (1) to hold, we would require $Pr(F(\mathbf{x}) \leq \theta_k | M_{k,B}) > Pr(F(\mathbf{x}) \leq \theta_k)$. given some relevant B. We will estimate the probability on the lhs from the sample generated using the model, and the probability on the rhs from the datasets provided. Conjecture (2) is related to the gain in recall obtained by using the model, although it is more practical to examine actual numbers of near-optimal instances (true-positives in the usual terminology). We will compare the numbers of near-optimal in the sample generated by the DBN model with ILP features, to those obtained using the DBN alone.

4.2 Materials

4.2.1 Data

We use two synthetic datasets, one arising from the KRK chess endgame (an endgame with just White King, White Rook and Black King on the board), and the other a restricted, but nevertheless hard 5×5 job-shop scheduling (scheduling 5 jobs taking varying lengths of time onto 5 machines, each capable of processing just one task at a time).

The optimisation problem we examine for the KRK endgame is to predict the depth-of-win with optimal play [1]. Although aspect of the endgame has not been as popular in ILP as task of predicting "White-to-move position is illegal" [2], it offers a number of advantages as a *Drosophila* for optimisation problems of the kind we are interested. First, as with other chess endgames, KRK-win is a complex, enumerable domain for which there is complete, noise-free data. Second, optimal "costs" are known for all data instances. Third, the problem has been studied by chess-experts at least since Torres y Quevado built a machine, in 1910, capable of playing the KRK endgame. This has resulted in a substantial amount of domain-specific knowledge. We direct the reader to [3] for the history of automated methods for the KRK-endgame. For us, it suffices to treat the problem as a form of optimisation, with the cost being the depth-of-win with Black-to-move, assuming minimax-optimal play. In principle, there are $64^3 \approx 260,000$ possible positions for the KRK endgame, not all legal. Removing illegal positions, and redundancies arising from symmetries of the board reduces the size of the instance space to about $28,000$ and the distribution shown in Fig. 4(a). The sampling task here is to generate instances with depth-of-win equal to 0. Simple random sampling has a probability of about $1/1000$ of generating such an instance once redundancies are removed.

The job-shop scheduling problem is less controlled than the chess endgame, but is nevertheless representative of many real-life applications (like scheduling trains), and in general, is known to be computationally hard.

Cost	Instances	Cost	Instances
0	27 (0.001)	9	1712 (0.196)
1	78 (0.004)	10	1985 (0.267)
2	246 (0.012)	11	2854 (0.368)
3	81 (0.152)	12	3597 (0.497)
4	198 (0.022)	13	4194 (0.646)
5	471 (0.039)	14	4553 (0.808)
6	592 (0.060)	15	2166 (0.886)
7	683 (0.084)	16	390 (0.899)
8	1433 (0.136)	draw	2796 (1.0)

Total Instances: 28056

(a) Chess

Cost	Instances	Cost	Instances
400–500	10 (0.0001)	1000–1100	24067 (0.748)
500–600	294 (0.003)	1100–1200	15913 (0.907)
600–700	2186 (0.025)	1200–1300	7025 (0.978)
700–800	7744 (0.102)	1300–1400	1818 (0.996)
800–900	16398 (0.266)	1400–1500	345 (0.999)
900–1000	24135 (0.508)	1500–1700	66 (1.0)

Total Instances: 100000

(b) Job-Shop

Figure 4: Distribution of cost values. The number in parentheses are cumulative frequencies.

Data instances for Chess are in the form of 6-tuples, representing the rank and file (X and Y values) of the 3 pieces involved. For the RBM, these are encoded as 48 dimensional binary vector where every eight bits represents a one hot encoding of the pieces' rank or file. At each iteration k of the EODS procedure, some instances with depth-of-win $\leq \theta_k$ and the rest with depth-of-win $> \theta_k$ are used to construct the ILP model, and the resulting features are appended to train the RBM model as described in Section 3.2.[3]

Data instances for Job-Shop are in the form of schedules, with associated start- and end-times for each task on a machine, along with the total cost of the schedule. On iteration i of the EODS procedure, models are to be constructed to predict if the cost of schedule will be $\leq \theta_i$ or otherwise.[4]

4.2.2 Background Knowledge

For Chess, background predicates encode the following (WK denotes, WR the White Rook, and BK the Black King): (a) Distance between pieces WK-BK, WK-BK, WK-WR; (b) File and distance patterns: WR-BK, WK-WR, WK-BK; (c) "Alignment distance": WR-BK; (d) Adjacency patterns: WK-WR, WK-BK, WR-BK; (e) "Between" patterns: WR between WK and BK, WK between WR and BK, BK between WK and WR; (f) Distance to closest edge: BK; (g) Distance to closest corner: BK; (h) Distance to centre: WK; and (i) Inter-piece patterns: Kings in opposition, Kings almost-in-opposition, L-shaped pattern. We direct the reader to [3] for the history of using these concepts, and their definitions. A sample rule generated for Depth<=2 is that the distance between the files of the two kings be greater than or equal to zero, and that the ranks of the kings are seperated bya a distance of less than five and those of the white king and the rook by less than 3.

For Job-Shop, background predicates encode: (a) schedule job J "early" on machine M (early means first or second); (b) schedule job J "late" on machine M (late means last or second-last); (c) job J has the fastest task for machine M; (d) job J has the slowest task for machine M; (e) job J has a fast task for machine M (fast means the fastest or second-fastest); (f) Job J has a slow task for machine M (slow means slowest or second-slowest); (g) Waiting time for machine M; (h) Total waiting time; (i) Time taken before executing a task on a machine. Correctly, the predicates for (g)–(i) encode upper and lower bounds on times, using the standard inequality predicates $\leq$ and $\geq$.

4.2.3 Algorithms and Machines

The ILP-engine we use is Aleph (Version 6, available from A.S. on request). All ILP theories were constructed on an Intel Core i7 laptop computer, using VMware virtual machine running Fedora 13, with an allocation of 2GB for the virtual machine. The Prolog compiler used was Yap, version 6.1.3[5]. The RBM was implemented in the Theano library, and run on an NVidia Tesla K-40 GPU Card.

[3]The θ_k values are pre-computed assuming optimum play. We note that when constructing a model on iteration k, it is permissible to use all instances used on iterations $1, 2, \ldots, (k-1)$ to obtain data for model-construction.

[4]The total cost of a schedule includes any idle-time, since for each job, a task before the next one can be started for that job. Again, on iteration i, it is permissible to use data from previous iterations.

[5]`http://www.dcc.fc.up.pt/~vsc/Yap/`

4.3 Method

Our method is straightforward:

> For each optimisation problem, and domain-knowledge B:
>
> Using a sequence of threshold values $\langle \theta_1, \theta_2, \ldots, \theta_n \rangle$ on iteration k ($1 \leq k \leq n$) for the EODS procedure:
>
> 1. Obtain an estimate of $Pr(F(\mathbf{x}) \leq \theta_k)$ using a DBN with a separator variable;
> 2. Obtain an estimate of $Pr(F(\mathbf{x}) \leq \theta_k | M_{k,B})$ by constructing an ILP model for discriminating between $F(\mathbf{x}) \leq \theta_k$ and $F(\mathbf{x}) > \theta_k$. Use the features learnt by the ILP model to guide the DBN sampling.
> 3. Compute the ratio of $Pr(F(\mathbf{x}) \leq \theta_k | M_{k,B})$ to $P(F(\mathbf{x}) \leq \theta_k)$

The following details are relevant:

- The sequence of thresholds for Chess are $\langle 8, 4, 2, 0 \rangle$. For Job-Shop, this sequence is $\langle 900, 890, 880 ... 600 \rangle$; Thus, θ^* = 0 for Chess and 600 for Job-Shop, which means we require exactly optimal solutions for Chess.
- Experience with the use of ILP engine used here (Aleph) suggests that the most sensitive parameter is the one defining a lower-bound on the precision of acceptable clauses (the $minacc$ setting in Aleph). We report experimental results obtained with $minacc = 0.7$, which has been used in previous experiments with the KRK dataset. The background knowledge for Job-Shop does not appear to be sufficiently powerful to allow the identification of good theories with short clauses. That is, the usual Aleph setting of upto 4 literals per clause leaves most of the training data ungeneralised. We therefore allow an upper-bound of upto 10 literals for Job-Shop, with a corresponding increase in the number of search nodes to 10000 (Chess uses the default setting of 4 and 5000 for these parameters).
- In the EODS procedure, the initial sample is obtained using a uniform distribution over all instances. Let us call this P_0. On the first iteration of EODS ($k = 1$), the datasets ${E_1}^+$ and ${E_1}^-$ are obtained by computing the (actual) costs for instances in P_0, and an ILP model $M_{1,B}$, or simply M_1, constructed. A DBN model is constructed both with and without ILP features. We obtained samples from the DBN with CD_6 or by running the Gibbs chain for six iterations. On each iteration k, an estimate of $Pr(F(\mathbf{x}) \leq \theta_k)$ can be obtained from the empirical frequency distribution of instances with values $\leq \theta_k$ and $> \theta_k$. For the synthetic problems here, these estimates are in Fig. 4. For $Pr(F(\mathbf{x}) \leq \theta_k | M_{k,B})$, we use obtain the frequency of $F(\mathbf{x}) \leq \theta_k$ in P_k
- Readers will recognise that the ratio of $Pr(F(\mathbf{x}) \leq \theta_k | M_{k,B})$ to $P(F(\mathbf{x}) \leq \theta_k)$ is equivalent to computing the gain in precision obtained by using an ILP model over a non-ILP model. Specifically, if this ratio is approximately 1, then there is no value in using the ILP model. The probabilities computed also provide one way of estimating sampling efficiency of the models (the higher the probability, the fewer samples will be needed to obtain an instance $\mathbf{x}$ with $F(\mathbf{x}) \leq \theta_k$).

4.4 Results

Results relevant to conjectures (1) and (2) are tabulated in Fig. 5 and Fig. 6. The principal conclusions that can drawn from the results are these:

(1) For both problems, and every threshold value θ_k, the probabilty of obtaining instances with cost at most θ_k with ILP-guided RBM sampling is substantially higher than without ILP. This provides evidence that ILP-guided DBN sampling results in better samples than DBN sampling alone(Conjecture 1);

(2) For both problems and every threshold value θ_k, samples obtained with ILP-guided sampling contain a substantially higher number of near-optimal instances than samples obtained using a DBN alone (Conjecture 2)

Additionally, Fig. 7 demonstrates the cumulative impact of ILP on (a) the distribution of good solutions obtained and (b)the cascading improvement over the DBN alone for the Job Shop problem. The DBN with ILP was able to arrive at the optimal solution within 10 iterations.

Model	$Pr(F(\mathbf{x}) \leq \theta_k \mid M_k)$			
	$k=1$	$k=2$	$k=3$	$k=4$
None	0.134	0.042	0.0008	0.0005
DBN	0.220	0.050	0.015	0.0008
DBNILP	0.345	0.111	0.101	0.0016

(a) Chess

Model	$Pr(F(\mathbf{x}) \leq \theta_k \mid M_k)$			
	$k=1$	$k=2$	$k=3$	$k=4$
None	0.040	0.036	0.029	0.024
DBN	0.209	0.234	0.248	0.264
DBNILP	0.256	0.259	0.268	0.296

(b) Job-Shop

Figure 5: Probabilities of obtaining good instances $\mathbf{x}$ for each iteration k of the EODS procedure. That is, the column $k = 1$ denotes $P(F(\mathbf{x}) \leq \theta_1$ after iteration 1; the column $k = 2$ denotes $P(F(\mathbf{x}) \leq \theta_2$ after iteration 2 and so on. In effect, this is an estimate of the precision when predicting $F(\mathbf{x}) \leq \theta_k$. "None" in the model column stands for probabilities of the instances, corresponding to simple random sampling ($M_k = \emptyset$).

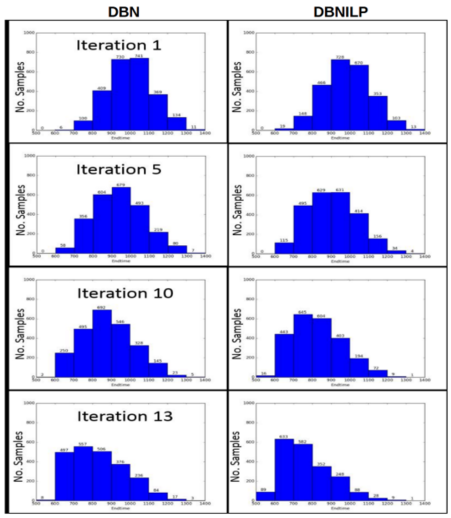
Model	Near-Optimal Instances			
	$k=1$	$k=2$	$k=3$	$k=4$
DBN	5/27	11/27	11/27	12/27
DBNILP	3/27	17/27	21/27	22/27

(a) Chess

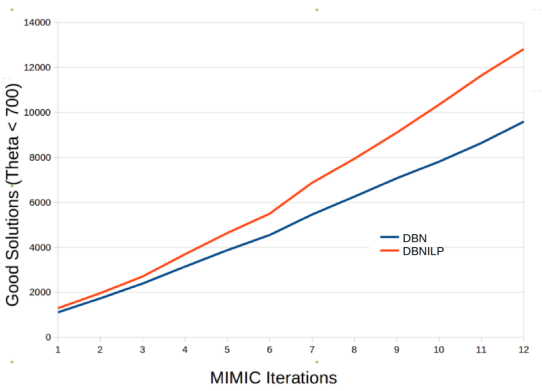
Model	Near-Optimal Instances		
	$k=11$	$k=12$	$k=13$
ILP	7/304	10/304	18/304
DBNILP	9/304	18/304	27/304

(b) Job-Shop

Figure 6: Fraction of near-optimal instances ($F(\mathbf{x}) \leq \theta^*$) generated on each iteration of EODS. In effect, this is an estimate of the recall (true-positive rate, or sensitivity) when predicting $F(\mathbf{x}) \leq \theta^*$. The fraction a/b denotes that a instances of b are generated.



(a)



(b)

Figure 7: Impact of ILP on EDOS procedure for Job Shop (a) Distribution of solution Endtimes generated on iterations 1, 5, 10 and 13 with and without ILP (b) Cumulative semi-optimal solutions obtained with and without ILP features over 13 iterations

5 Conclusions and Future Work

In this paper we demonstrate that DBNs can be used as efficient samplers for EDA style optimization approaches. We further look at combining the sampling and feature discovery power of Deep Belief Networks with the background knowledge discovered by an ILP engine, with a view towards optimization problems that entail some degree of domain information. The optimization is performed iteratively via an EDA mechanism and empirical results demonstrate the value of incorporating ILP based features into the DBN. In the future we intend to combine ILP based background rules with more sophisticated deep generative models proposed recently [5, 6] and look at incorporating the rules directly into the cost function as in [8].

References

[1] M. Bain and S. Muggleton. Learning optimal chess strategies. In K. Furukawa, D. Michie, and S. Muggleton, editors, *Machine Intelligence 13*, pages 291–309. Oxford University Press, Inc., New York, NY, USA, 1995.

[2] Michael Bain. Learning logical exceptions in chess, 1994. PhD Thesis, University of Strathclyde.

[3] Gabriel Breda. KRK Chess Endgame Database Knowledge Extraction and Compression, 2006. Diploma Thesis, Technische Universität, Darmstadt.

[4] Jeremy S De Bonet, Charles L Isbell, Paul Viola, et al. Mimic: Finding optima by estimating probability densities. *Advances in neural information processing systems*, pages 424–430, 1997.

[5] Danihelka I. Mnih A. Blundell C. Gregor, K. and D. Wierstra. Deep autoregressive networks. In *ICML*, 2014.

[6] Danihelka I. Mnih A. Blundell C. Gregor, K. and D. Wierstra. D. draw: A recurrent neural network for image generation. In *ICML*, 2015.

[7] Geoffrey Hinton. Deep belief nets. In *Encyclopedia of Machine Learning*, pages 267–269. Springer, 2011.

[8] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*, 2016.

[9] Sachindra Joshi, Ganesh Ramakrishnan, and Ashwin Srinivasan. Feature construction using theory-guided sampling and randomised search. In *ILP*, pages 140–157, 2008.

[10] Ganesh Ramakrishnan, Sachindra Joshi, Sreeram Balakrishnan, and Ashwin Srinivasan. Using ILP to construct features for information extraction from semi-structured text. In *ILP*, pages 211–224, 2007.

[11] Amrita Saha, Ashwin Srinivasan, and Ganesh Ramakrishnan. What kinds of relational features are useful for statistical learning? In *ILP*, 2012.

[12] Lucia Specia, Ashwin Srinivasan, Sachindra Joshi, Ganesh Ramakrishnan, and Maria Graças Volpe Nunes. An investigation into feature construction to assist word sense disambiguation. *Machine Learning*, 76(1):109–136, 2009.

[13] Lucia Specia, Ashwin Srinivasan, Ganesh Ramakrishnan, and Maria das Graças Volpe Nunes. Word sense disambiguation using inductive logic programming. In *ILP*, pages 409–423, 2006.

[14] Ashwin Srinivasan, Gautam Shroff, Lovekesh Vig, Sarmimala Saikia, and Puneet Agarwal. Generation of near-optimal solutions using ilp-guided sampling. In *ILP*. 2016.

[15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

Top-Down and Bottom-Up Interactions between Low-Level Reactive Control and Symbolic Rule Learning in Embodied Agents

Clément Moulin-Frier
SPECS Lab
Universitat Pompeu Fabra
Barcelona, Spain
clement.moulinfrier@gmail.com

Xerxes D. Arsiwalla
SPECS Lab
Universitat Pompeu Fabra
Barcelona, Spain
x.d.arsiwalla@gmail.com

Jordi-Ysard Puigbò
SPECS Lab
Universitat Pompeu Fabra
Barcelona, Spain
jordiysard.puigbo@upf.edu

Martì Sanchez-Fibla
SPECS Lab
Universitat Pompeu Fabra
Barcelona, Spain
santmarti@gmail.com

Armin Duff
SPECS Lab
Universitat Pompeu Fabra
Barcelona, Spain
armin.duff@gmail.com

Paul FMJ Verschure
SPECS Lab
Universitat Pompeu Fabra & ICREA
Barcelona, Spain
paul.verschure@upf.edu

Abstract

Mammals bootstrap their cognitive structures through embodied interaction with the world. This raises the question: how the reactive control of initial behavior is recurrently coupled to the high-level symbolic representations they give rise to? We investigate this question in the framework of the "Distributed Adaptive Control" (DAC) cognitive architecture, where we study top-down and bottom-up interactions between low-level reactive control and symbolic rule learning in embodied agents. Reactive behaviors are modeled using a neural allostatic controller, whereas high-level behaviors are modeled using a biologically-grounded memory network reflecting the role of the prefrontal cortex. The interaction of these modules in a closed-loop fashion suggests how symbolic representations might have been shaped from low-level behaviors and recruited for behavior optimization.

1 Introduction

A major challenge in cognitive neuroscience is to propose a unified theory of cognition based on general models of the brain structure [20]. Such a theory should be able to explain how specific cognitive functions (e.g. decision making or planning) result from the particular dynamics of an embodied cognitive architecture in a specific environment. This has led to various proposals, formalizing how cognition arises from interaction of functional modules. Early implementations of cognitive architectures trace back to the era of Symbolic Artificial Intelligence, starting from the General Problem Solver (GPS, [21]) and has been followed by a number of subsequent architectures such as Soar [13, 12], ACT-R [1, 2] and their follow-ups. These architectures are considered top-down and representation-based in the sense that they consist of a complex representation of a task, which

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

has to be decomposed recursively into simpler ones to be executed by the agent. Although relatively powerful at solving abstract symbolic tasks, top-down architectures have enjoyed very little success at bootstrapping behavioral processes and taking advantage of the agent's embodiment (nonetheless, several interfaces with robotic embodiment have been proposed, see [28, 24]). In contrast to top-down representation-based approaches, behavior-based robotics [7] emphasizes lower-level sensory-motor control loops as a starting point of behavioral complexity that can be further extended by combining multiple control loops together, e.g. as in the subsumption architecture [6]. Such approaches are also known as bottom-up and they generally model behavior without relying on complex knowledge representation and reasoning. This is a significant departure from Newell's and Anderson's views of cognition expressed in GPS, Soar and ACT-R. Top-down and bottom-up approaches thus reflect different aspects of cognition: high-level symbolic reasoning for the former and low-level embodied behaviors for the latter. However, both aspects are of equal importance when it comes to defining a unified theory of cognition. It is therefore a major challenge of cognitive science to unify both approaches into a single theory, where (a) reactive control allows an initial level of complexity in the interaction between an embodied agent and its environment and (b) this interaction provides the basis for learning higher-level symbolic representations and for sequencing them in a causal way for top-down goal-oriented control. We propose to split the problem of how neural and symbolic approaches are integrated into the following three research questions:

- How are high-level symbolic representations shaped from low-level reactive behaviors in a bottom-up manner?
- How are those representations recruited in rule and plan learning?
- How do rules and plans modulate reactive behaviors through top-down control for realizing long-term goals?

To address these questions, we adopt the principles of the *Distributed Adaptive Control* (DAC) theory of the mind and brain [31, 30], which posits that cognition is based on the interaction of four interconnected control loops operating at different levels of abstraction (Fig. 1). The first level is the embodiment of the agent within its environment, with the sensors and actuators of the agent (called the Somatic layer). The Somatic layer incorporates physiological needs of the agent (e.g. exploration or safety) and which drives the dynamics of the whole architecture [23]. Extending behavior-based approaches with drive reduction mechanisms, complex behavior is bootstrapped in DAC from the self-regulation of an agent's physiological needs when combined with reactive behaviors (the Reactive layer). This reactive interaction with the environment drives learning processes for acquiring a state space of the agent-environment interaction (the Adaptive layer) and the acquisition of higher-level cognitive abilities such as abstract goal selection, memory and planning (the Contextual layer). These high-level representations in turn modulate behavior at lower levels via top-down pathways shaped by behavioral feedback. The control flow in DAC is therefore distributed, both from bottom-up and top-down interactions between layers, as well as from lateral information processing into the subsequent layers.

In this paper, we present biologically-grounded neural models of the reactive and contextual layers and discuss their possible integration to address the question of how the reactive control of initial behavior is recurrently coupled to the high-level symbolic representations they give rise to. On one hand, our model of the reactive layer relies on the concept of allostatic control. Sterling proposes that allostasis drives regulation through anticipation of needs [27]. In [26], allostasis is seen as a reactive meta-regulation system of homeostatic loops that are possibly contradicting each other in a winner-takes-all process, modulated by emotional and physiological feedback. On the other hand, we present a neural model of the contextual layer for rule and plan learning grounded in the neurobiology of the prefrontal cortex (PFC) [8]. It has been shown that representations of sensory states, actions and their combinations can be found in the PFC [16, 9] and that is reciprocally connected to sensory as well as motor areas [9]. This puts the PFC in a favorable position for the representation of sensory-motor contingencies, i.e. patterns of sensory-motor dependencies [22], selected according to their relevance in goal-oriented behavior. Its involvement in flexible cognitive control and planning (e.g. [10, 5]) backs the hypothesis that sensory-motor contingencies promote flexible cognitive control and planning. This paper aims at identifying the key neurocomputational challenges in integrating both models in a complete cognitive architecture.

The next section introduces a model of the reactive layer based on the concept of allostatic control and is concerned with prioritization of multiple self-regulation loops. We then present an existing model [8] of the prefrontal cortex that is able to integrate sensory-motor contingencies in rules and plans for long-term reward maximization. Finally, we discuss how both models can be integrated to

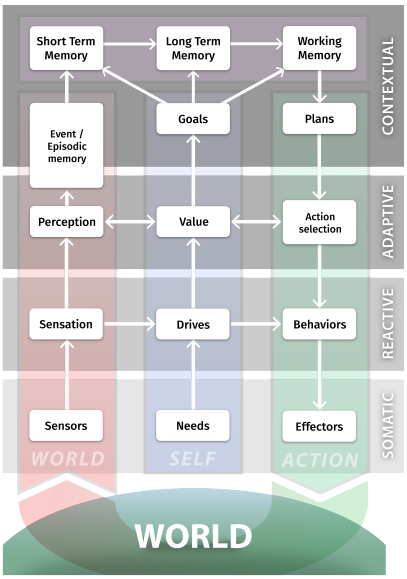


Figure 1: DAC proposes that cognition is organized as a layered control structure with tight coupling within and between these layers (adapted from [30]): the Somatic, Reactive, Adaptive and Contextual layers. Across these layers, a columnar organization exists that deals with the processing of states of the World or exteroception (left, red), the Self or interoception (middle, blue) and Action (right, green). The role of each layer and their interaction is described in the text.

bridge the gap between low-level reactive control and high-level symbolic rule learning in embodied agents. We will make a particular emphasis on how the acquired rules are able to inhibit the reactive system to achieve long-term goals as proposed in theories on the neuropsychology of anxiety [11] and consciousness [29, 3, 4].

2 A neural model of allostatic control

In this section we describe the concept of allostatic control and introduces a neural model for it. Based on the principles of DAC, we consider an embodied agent endowed with physiological needs for e.g. foraging or safety and self-regulating them through parallel drive reduction control loops. Drives aim at self-regulating internal state variables within their respective homeostatic ranges. Such an internal state variable could, for example, reflect the current glucose level in an organism, with the associated homeostatic range defining the minimum and maximum values of that level. A drive for foraging would then correspond to a self-regulatory mechanism where the agent actively searches for food whenever its glucose level is below the homeostatic minimum, and stops eating even if food is present whenever levels are above the homeostatic maximum. A drive is therefore defined as the real-time control loop triggering appropriate behaviors whenever the associated internal state variable goes out of its homeostatic range, as a way to self-regulate its value in a dynamic and autonomous way. It is common for drives to be competing and we thus require a method to prioritize them. Consider for example a child in front of a transparent box full of candies, with a caregiver telling her to not open the box. Two drives are competing in such a situation: one for eating candies and one for obeying to the caregiver. Depending on how fond of candies she is and of how strict the caregiver is, she will choose to either break a social rule or to enjoy an extremely pleasant moment. Such regulation conflicts can be solved through the concept of an allostatic controller [26], defined as a set of parallel homeostatic control loops operating in real-time and dealing with their prioritization to ensure an efficient global regulation of multiple internal state variables.

The allostatic control model we introduce in this paper is composed of three subsystems (Fig. 2). Motor primitive neurons are connected to the agent actuators through synaptic connections. For example, on a 2-wheeled mobile robot, a "turn right" motor primitive would excite the left wheel actuator and inhibit the right wheel one, whereas a "go forward" motor primitive would excite

both. A repertoire of behaviors (inner boxes inside the middle box) link sensation neurons (middle nodes in the behavior boxes) to motor primitive neurons (right nodes in the behavior boxes) through behavior-specific connections. In a foraging behavior for example, sensing food on the right would activate a "turn right" motor primitive, whereas in an obstacle avoidance action, sensing an obstacle on the right would activate a "turn left" motor primitive. In Figure 2, two behaviors are connected to the two same motor primitives. In the general case however, they could only connect to a specific relevant subset of the motor primitives. The agent's exteroceptive sensors (vertical red bar on the middle) are connected to sensation nodes in the behavior subsystem (middle nodes in each inner box) by connections not shown in the figure. Each behavior is provided with an input activation node (left node in each behavior box) which has a baseline activity (indicated by the number 1) inhibiting the sensation nodes. Therefore, if no input is provided to an activation node, the sensation nodes of the corresponding behavior are inhibited, preventing information to propagate to the motor primitives. Drive nodes (left) can activate their associated behavior through the inhibition of the corresponding activation nodes, that in turn disinhibit sensation nodes through a double inhibition process, allowing sensory information to propagate to the motor primitives. In Figure 2, two drives nodes are represented, each connected to its associated behavior which is supposed to reduce the drive activity (e.g. a foraging drive connecting to food attraction behavior). Drive nodes are activated through connections from the agent's interoceptive sensors (vertical red bar on the left) and form a winner-takes-all process through mutual inhibition. This way, drives compete against each other as in the child-caregiver example presented above.

3 Contextual layer: a neural implementation of a rule learning system

In the context of the DAC architecture, we use an existing biologically-grounded model for rule learning and planning developed in our research group [8] based on key physiological properties of the prefrontal cortex (PFC), i.e. reward modulated sustained activity and plasticity of lateral connectivity. Recent proposals highlight the role of sensory–motor contingencies as the building blocks of many intelligent behaviors including rule learning and planning. Sensory–motor contingencies combine information about perceptual inputs and related motor actions forming internal states, which are subsequently used to structure and plan behavior. The DAC architecture has been developed to investigate how sensory–motor contingencies can be formed and exploited for behavioral control such as rule learning and flexible planning. Contingencies formed at the level of the adaptive layer provide inputs to the contextual layer, which acquires, retains, and expresses sequential representations using systems for short-term and long-term memory. The PFC-grounded contextual layer consists of a group of laterally connected memory-units. Each memory-unit is selective for one specific stimulus, e.g. color, and can induce one specific action, e.g. "go left", "go forward" or "go right", forming a sensory–motor contingency (Fig. 3). A memory-unit can be interpreted as a micro-column comprising a number of neurons with the same coding properties. Sequential rules are expressed through the coordinated activation of different memory-units in the correct order. This group of memory-units forms the elementary substrate for the representation and expressions of rules.

The activity of memory-units in this architecture is driven by perceptual inputs, observed reward and state prediction through the lateral connectivity. Memory-units compete in a probabilistic selection mechanism. The higher the activity, the higher the probability of being selected. The selected memory-units propagate their activity and contribute to the final action of the agent. To express a rule, the activity of these memory-units must be modulated in order to control the selection of specific units contributing to the final action. The modulated activity of these units is influenced by two systems, the lateral connectivity and the reward system. The lateral connectivity captures the context and the order of the sequential rules and influences activity through trigger values. Trigger values allow to chain through a specific sequence of memory-units. The reward system validates different rules represented in the network and influences the activity through the reward value. The modulation of each memory-unit activity is realized by multiplying the perceptual activity by the trigger value as well as the reward value. This model has been validated in simulated robotic experiments where stimuli-response associations have to be learned to maximize reward in a multiple T-maze environment, as well in solving the Towers of London task. Moreover the model is able to re-adapt to a changing environment, e.g. when the stimuli-response associations are modified on-line [8].

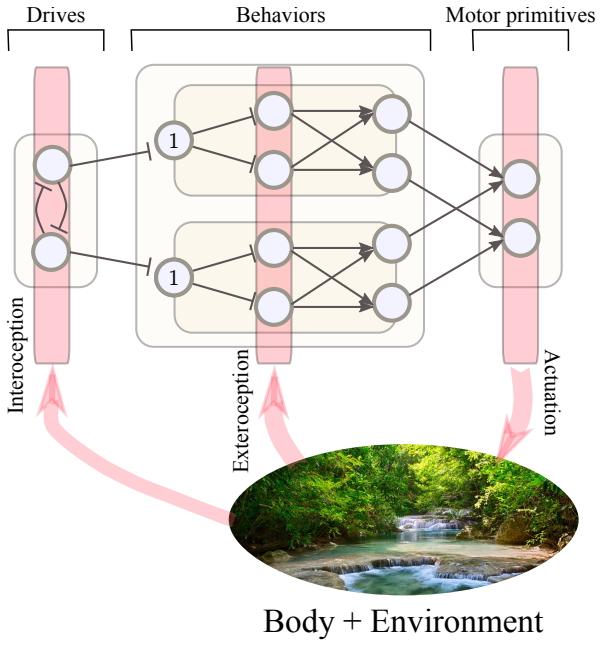


Figure 2: The allostatic control model we introduce in this paper is composed of three subsystems: Drives, Behaviors and Motor Primitives, represented by the left, middle and right boxes, respectively. The behavior subsystem is itself composed of multiple behaviors represented by the inner boxes (two behaviors are represented in the figure). Inside those subsystems, round nodes represent neurons that can have a baseline activation (indicated by a number 1) or not (no number). Synaptic connections between neurons are represented by dark arrows, where a peak end (resp. a bar end) indicates an excitatory (resp. inhibitory) connection. Vertical red bars represent neural populations that act as the interface of the agent with its environments through interoceptive sensations (left), exteroceptive sensations (middle) and actuation (right). Those neural populations interact in a predefined way with the neuron nodes they overlap with, trough connections not shown in the figure. On a 2-wheeled mobile robot for example, motor primitives for e.g. turning right or going forward would be represented by nodes in the motor primitive subsystem and connected to the left and right wheels actuators represented by the vertical red bar on the right. Similarly, left and right proximeter sensors would be represented by the middle vertical bar and connected to the sensory neurons of the behaviors that require this information, as it would be the case of the sensory neurons of an obstacle avoidance behavior but not of a light following one (because the latter behavior doesn't require proximeter information).

4 Bridging the gap between reactive control with continuous sensory-motor contingencies and symbolic prediction through rule learning

We have presented in the two last sections two neural models.
• The first, that we call the *reactive layer*, implements an allostatic controller which deals with the parallel execution of possibly conflicting self-regulation control loops in real-time. It is composed of three subsystems: *Drives* that are activated through interoception (related e.g. to the organism's glucose level for a foraging drive), *Behaviors* implemented by reactive control loops (e.g. attraction to food spot when the glucose level is low) and activation of *Motor Primitives* that send the required commands to the agent's actuators (e.g. turning left or turning right).
• The second, that we call the *contextual layer*, implements a rule learning system composed of a large number of memory units that each encodes a specific sensory-motor contingency, i.e. is sensitive to a particular sensory receptive field and activates a particular action. Lateral connectivity between units, learned from the agent's experience, allows the temporal chaining of sensory-motor contingencies through action. Each unit is associated with a reward value that is also learned and predict the (possibly delayed) reward expected by executing the corresponding action encoded by the unit in the corresponding sensory state it is selective to. At a given time, the activity of each memory

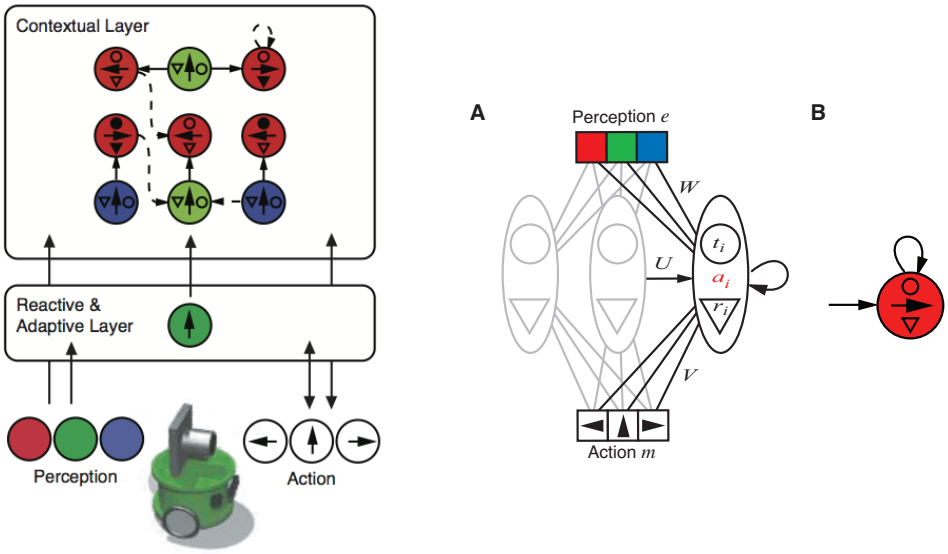


Figure 3: Left: In the DAC architecture the reactive/adaptive layer provides the agent with simple automatic behaviors and forms sensory–motor contingencies used in the contextual layer. Each circle stands for a bi-modal perception–action memory unit. The color of the circle indicates the perception selectivity while the arrow indicates the action specificity. The contextual layer is driven by the sensory input. The lateral connectivity primes the sequence of the activation. A reward-value system modifies the activity in order to select between the memory-units primed by the lateral connectivity. An action selection mechanism provides the final action for the agent. Right: Implementation of a single memory-unit (in inset A). The perceptual selectivity a_i is driven by perception e through the weight matrix W. The trigger value t_i is driven by the activation of other memory-units through the (learned) weights U. The reward value is noted r_i. The motor activity induced by a memory-unit is driven by its total activity $a_i * t_i * r_i$ through the weight matrix V. Inset B: symbolic representation of the same memory-unit, where the color represents the perceptual selectivity, the inner arrow the motor specificity and the external arrows the lateral connectivity.

unit is driven by the sensitivity of that unit to the current agent's perception, the prediction of that perception from the lateral connections and the reward value learned by the unit over time. The most activated units then compete together to decide what next action should be performed to maximize future rewards.

In this section, we discuss how both layers can be integrated in a complete cognitive architecture where contextual rules are learned from sensory-motor actions generated by reactive control, and where the actions generated by the contextual layer in turn, modulates the activity of the reactive system to achieve long-term goals. To illustrate the cognitive dynamics at work, let's consider again the example we have described above about a child that experiences a conflict between eating candies and disobeying to a caregiver. How the reactive and contextual layer models we have presented in the two last sections interact together in such a situation? Without any previous experience of it, only the reactive layer generates behavior through the self-regulation of internal drives, here eating candies and obeying the caregiver. Satisfying one or the other drive depends on their respective current level as well as the parameters of their mutual inhibition (one could conceive it as personality, rather a rebel or a good child). Since this situation occurs in various contexts, where the drive initial activities differ, the consequences of both behaviors are experienced by the child. When she obeys the caregiver, she is frustrated of not eating a candy. When she disobeys, she has an argument with the caregiver and likely doesn't eat any candy anyway according to how strict her opponent is. By experiencing multiple times the consequences of the two actions, she will discover that respecting

the social rule is of better interest to maximize reward, consequently self-inhibiting her own drive for eating candies by contextual top-down control acting on the reactive layer. Besides this real-life example, such a top-down "behavioral inhibition system" has been proposed as a major component of theories on the neuropsychology of consciousness [29] and anxiety [11].

A computational implementation of the reactive and contextual layer integration has to solve the issues of (a) preprocessing sensory-motor information generated by the reactive layer to provide relatively abstract and stable perceptions of the environment at the contextual level, (b) modulating the memory-unit activities through reward values derived from the drive levels, (c) modulating the activity of drive levels from the output generated by the contextual layer to maximize reward.

Solving (a) requires the addition of an adaptive layer in the architecture that acquires a state space of the agent-environment interaction through perceptual learning mechanism modulated by reward. In previous computational models of the DAC architecture, this is accounted as associative learning [14]. In the context of the two models presented in this paper, a solution consists of a direct connection from the sensation neurons of the reactive layer to the perceptual activation of memory units as in Fig. 3. In the current version of the rule learning model, a large number of memory-units is generated with pseudo-random perceptual sensitivity and motor specificity. Therefore the total number of units has to be large compared to the number of units that are actually recruited for generating actions after learning. Adaptively tuning memory-unit sensitivity would allow optimization of the size of the network, e.g. through a learning rule able to attract perceptual sensitivity of memory units according to their rewarding effect. Note that a similar learning rule can be applied for learning the motor specificity of memory-units. Besides a direct connection between the neural units of both layers, neural hierarchies would also improve rule learning by providing more abstract information to memory-units. This can be achieved using recent advances in Deep Reinforcement Learning, as e.g. in [17], able to learn highly abstract perceptual representations from raw sensory data driven by reward; or by adopting a more biologically-grounded approach using neurocomputational models of perceptual learning grounded in the neurobiology of the cerebral cortex interaction with the amygdala [25]. Neuromodulators independent from reward are also present during aversive events, sustained attention or surprise, modulating not just memory formation but perception as well. The temporal stabilization of sensory features have been proposed in [32]. Note that not only sensory information can be abstracted and stabilized, but hierarchies of needs [15] and behaviors as well.

To solve (b), reward values can directly be computed from drive activities: the lower the drive activity the better it is satisfied. Drive-related activities can also modulate the reward value of memory units according to the physiological context. This way, lateral connectivity among memory units is coding for the causal link between sensory-motor contingencies, supposed to be context-independent, whereas reward activation depends on the current configuration of drive levels (one could call it the emotional state of the agent). This will allow the memory network to behave over different rule sets according to the internal state of the agent (the rules maximizing reward when an agent is hungry are not the same as when it is sleepy), avoiding learning interference between different situations.

Finally, regarding (c), we propose that the contextual layer's activity modulates the reactive one by directly acting on the drive levels, instead of acting latter in the reactive layer pipeline, e.g. on the motor primitives. By directly modulating the drive levels, the contextual layer takes full control of the reactive one by acting on it from the source (see Fig. 2. This allows the agent to self-inhibit some of its own drives for maximizing reward on the long term as in the child-caregiver example.

5 Conclusion

In this paper we have designed an allostatic controller and introduced a novel computational model implementing it. This reactive layer allows the self-regulation of multiple drive-reduction control loop operating in parallel. Then we have presented an existing computational model of a contextual layer grounded in the neurobiology of the prefrontal cortex and able to learn sequential rules and plans through experience. Finally, our main contribution in this paper has been to argue for both a bottom-up and top-down interaction between low-level reactive control and high-level contextual plans. Symbolic representations in our approach are learned as sensory-motor contingencies encoded in discrete memory-units from the activity generated by the reactive layer. In turn, the actions generated by the contextual layer modulates the activity of the reactive system through a top-down pathway, inhibiting reactive drives to achieve long-term goals.

Both the reactive and contextual models are implemented and we are now working on their computational integration. We have identified in the last section the main challenges in term of bottom-up perceptual abstraction, multitask reward optimization as well as top-down drive modulation. This will allow a computational implementation of a reactive agent, embodied in a physical mobile robot and progressively acquiring contextual rules of its environment from experience, thus demonstrating an increasingly rational behavior.

We will also put a particular emphasis on applying this integrated cognitive architecture to study the formation of social norms in multi-robot setups [19, 18]. The long term goal is to understand how the constraints imposed by a multi-agent environment favor conscious experience. The research direction we adopt is based on the hypothesis that social norms are needed for the evolution of large multi-agent groups and that the formation of those social norms requires each individual to take conscious control of its own drive system [29].

Acknowledgments

Work supported by ERC's CDAC project:"Role of Consciousness in Adaptive Behavior" (ERC-2013-ADG 341196); & EU projects *Socialising Sensori-Motor Contingencies* socSMC-641321—H2020-FETPROACT-2014 & *What You Say Is What You Did* WYSIWYD (FP7 ICT 612139).

References

[1] J. R. Anderson. *The Architecture of Cognition*. Harvard University Press, 1983.

[2] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036–1060, 2004.

[3] X. D. Arsiwalla, I. Herreros, C. Moulin-Frier, M. Sanchez, and P. F. Verschure. Is consciousness a control process? In *International Conference of the Catalan Association for Artificial Intelligence*, pages 233–238. IOS, 2016.

[4] X. D. Arsiwalla, I. Herreros, and P. Verschure. On three categories of conscious machines. In *Conference on Biomimetic and Biohybrid Systems*, pages 389–392. Springer, 2016.

[5] W. F. Asaad, G. Rainer, and E. K. Miller. Task-specific neural activity in the primate prefrontal cortex. *Journal of Neurophysiology*, 84(1):451–459, 2000.

[6] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

[7] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47(1-3):139–159, 1991.

[8] A. Duff, M. S. Fibla, and P. F. Verschure. A biologically based model for the integration of sensory–motor contingencies in rules and plans: A prefrontal cortex based extension of the distributed adaptive control architecture. *Brain research bulletin*, 85(5):289–304, 2011.

[9] J. Fuster. *The Prefrontal Cortex: Anatomy, Physiology, and Neurophysiology of the Frontal Lobe*. Lippincott-William & Wilkins, Philadelphia, 1997.

[10] J. M. Fuster, G. E. Alexander, et al. Neuron activity related to short-term memory. *Science*, 173(3997):652–654, 1971.

[11] J. A. Gray and N. McNaughton. *The neuropsychology of anxiety: An enquiry into the function of the septo-hippocampal system*. Number 33. Oxford university press, 2003.

[12] J. E. Laird. *The Soar Cognitive Architecture*. MIT Press, 2012.

[13] J. E. Laird, A. Newell, and P. S. Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[14] E. Marcos, M. Ringwald, A. Duff, M. Sánchez-Fibla, and P. F. Verschure. The hierarchical accumulation of knowledge in the distributed adaptive control architecture. In *Computational and robotic models of the hierarchical organization of behavior*, pages 213–234. Springer, 2013.

[15] A. Maslow. A theory of human motivation. *Psychological review*, 1943.

[16] E. K. Miller, L. Li, and R. Desimone. Activity of neurons in anterior inferior temporal cortex during a short-term memory task. *The Journal of Neuroscience*, 13(4):1460–1478, 1993.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[18] C. Moulin-Frier, M. Sanchez-Fibla, and P. F. Verschure. Autonomous development of turn-taking behaviors in agent populations: a computational study. In *IEEE International Conference on Development and Learning, ICDL/Epirob, Providence (RI), USA*, 2015.

[19] C. Moulin-Frier and P. F. Verschure. Two possible driving forces supporting the evolution of animal communication. comment on" towards a computational comparative neuroprimatology: Framing the language-ready brain" by michael a. arbib. *Physics of life reviews*, 16:88–90, 2016.

[20] A. Newell. *Unified theories of cognition*. Harvard University Press, 1990.

[21] A. Newell, J. C. Shaw, and H. A. Simon. Report on a general problem-solving program. *IFIP Congress*, pages 256–264, 1959.

[22] J. K. O'Regan and A. Noë. A sensorimotor account of vision and visual consciousness. *The Behavioral and brain sciences*, 24(5):939–73; discussion 973–1031, oct 2001.

[23] J.-Y. Puigbò, C. Moulin-Frier, and P. F. Verschure. Towards self-controlled robots through distributed adaptive control. In *Conference on Biomimetic and Biohybrid Systems*, pages 490–497. Springer, 2016.

[24] J.-Y. Puigbo, A. Pumarola, C. Angulo, and R. Tellez. Using a cognitive architecture for general purpose service robot control. *Connection Science*, 27(2):105–117, 2015.

[25] J.-Y. Puigbò, G. Maffei, M. Ceresa, G.-B. M.A., and V. P.F.M.J. Learning relevant features through a two-phase model of conditioning. *IBM Journal of Research and Development, Special issue on Computational Neuroscience*, Accepted, in Press.

[26] M. Sanchez-Fibla, U. Bernardet, E. Wasserman, T. Pelc, M. Mintz, J. C. Jackson, C. Lansink, C. Pennartz, and P. F. Verschure. Allostatic control for robot behavior regulation: a comparative rodent-robot study. *Advances in Complex Systems*, 13(03):377–403, 2010.

[27] P. Sterling. Allostasis: a model of predictive regulation. *Physiology & behavior*, 106(1):5–15, 2012.

[28] G. Trafton, L. Hiatt, A. Harrison, F. Tamborello, S. Khemlani, and A. Schultz. ACT-R/E: An Embodied Cognitive Architecture for Human-Robot Interaction. *Journal of Human-Robot Interaction*, 2(1):30–55, Mar 2013.

[29] P. F. Verschure. Synthetic consciousness: the distributed adaptive control perspective. *Phil. Trans. R. Soc. B*, 371(1701):20150448, 2016.

[30] P. F. M. J. Verschure, C. M. A. Pennartz, and G. Pezzulo. The why, what, where, when and how of goal-directed choice: neuronal and computational principles. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 369(1655):20130483, 2014.

[31] P. F. M. J. Verschure, T. Voegtlin, and R. J. Douglas. Environmentally mediated synergy between perception and behaviour in mobile robots. *Nature*, 425(6958):620–624, 2003.

[32] Y. Yamashita and J. Tani. Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Comput Biol*, 4(11):e1000220, 2008.

MS MARCO: A Human Generated MAchine Reading COmprehension Dataset

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary,
Rangan Majumder and Li Deng
Microsoft AI & Research
Bellevue, WA, USA
{trnguye, miriamr, xiaso, jfgao, satiwary, ranganm, deng}@microsoft.com

Abstract

This paper presents our recent work on the design and development of a new, large scale dataset, which we name MS MARCO, for MAchine Reading COmprehension. This new dataset is aimed to overcome a number of well-known weaknesses of previous publicly available datasets for the same task of reading comprehension and question answering. In MS MARCO, all questions are sampled from real anonymized user queries. The context passages, from which answers in the dataset are derived, are extracted from real web documents using the most advanced version of the Bing search engine. The answers to the queries are human generated. Finally, a subset of these queries has multiple answers. We aim to release one million queries and the corresponding answers in the dataset, which, to the best of our knowledge, is the most comprehensive real-world dataset of its kind in both quantity and quality. We are currently releasing 100,000 queries with their corresponding answers to inspire work in reading comprehension and question answering along with gathering feedback from the research community.

1 Introduction

Building intelligent agents with the ability for reading comprehension (RC) or open-domain question answering (QA) over real world data is a major goal of artificial intelligence. Such agents can have tremendous value for consumers because they can power personal assistants such as Cortana [3], Siri [6], Alexa [1], or Google Assistant [4] found on phones or headless devices like Amazon Echo [2], all of which have been facilitated by recent advances in deep speech recognition technology [18, 9]. As these types of assistants rise in popularity, consumers are finding it more convenient to ask a question and quickly get an answer through voice assistance as opposed to navigating through a search engine result page and web browser. Intelligent agents with RC and QA abilities can also have incredible business value by powering bots that automate customer service agents for business found through messaging or chat interfaces.

Real world RC and QA is an extremely challenging undertaking involving the amalgamation of multiple difficult tasks such as reading, processing, comprehending, inferencing/reasoning, and finally summarizing the answer.

The public availability of large datasets has led to many breakthroughs in AI research. One of the best examples is ImageNet's [10] exceptional release of 1.5 million labeled examples and 1000 object categories which has led to better than human level performance on object classification from images [15]. Another example is the very large speech databases collected over 20 years by DARPA that enabled successes of deep learning in speech recognition [11]. Recently there has been an influx of datasets for RC and QA as well. These databases, however, all have notable drawbacks. For example, some are not large enough to train deep models [27], and others are larger but are synthetic.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

One characteristic in most, if not all, of the existing databases for RC and QA research is that the distribution of questions asked in the databases are not from real users. In the creation of most RC or QA datasets, usually crowd workers are asked to create questions for a given piece of text or document. We have found that the distribution of actual questions users ask intelligent agents can be very different from those conceived from crowdsourcing them from the text.

Furthermore, real-world questions can be messy: they may include typos and abbreviations. Another characteristic of current datasets is that text is often from high-quality stories or content such as Wikipedia. Again, real-world text may have noisy or even conflicting content across multiple documents and our experience is that intelligent agents will often need to operate over this type of problematic data.

Finally, another unrealistic characteristic of current datasets is that answers are often restricted to an entity or a span from the existing reading text. What makes QA difficult in the real world is that an existing entity or a span of text may not be sufficient to answer the question. Finding the best answer as the output of QA systems may require reasoning across multiple pieces of text/passages. Users also prefer answers that can be read in a stand-alone fashion; this sometimes means stitching together information from multiple passages, as the ideal output not only answers the question, but also has supporting information or an explanation.

In this paper we introduce Microsoft MAchine Reading Comprehension (MS MARCO) - a large scale real-world reading comprehension dataset that addresses the shortcomings of the existing datasets for RC and QA discussed above. The questions in the dataset are real anonymized queries issued through Bing or Cortana and the documents are related web pages which may or may not be enough to answer the question. For every question in the dataset, we have asked a crowdsourced worker to answer it, if they can, and to mark relevant passages which provide supporting information for the answer. If they can't answer it we consider the question unanswerable and we also include a sample of those in MS MARCO. We believe a characteristic of reading comprehension is to understand when there is not enough information or even conflicting information so a question is unanswerable. The answer is strongly encouraged to be in the form of a complete sentence, so the workers may write a longform passage on their own. MS MARCO includes 100,000 questions, 1 million passages, and links to over 200,000 documents. Compared to previous publicly available datasets, this dataset is unique in the sense that (a) all questions are real user queries, (b) the context passages, which answers are derived from, are extracted from real web documents, (c) all the answers to the queries are human generated, (d) a subset of these queries has multiple answers, (e) all queries are tagged with segment information.

2 Related Work

Dataset	Segment	Query Source	Answer	# Queries	# Documents
MCTest	N	Crowdsourced	Multiple choice	2640	660
WikiQA	N	User logs	Sentence selection	3047	29.26K sentences
CNN/Daily Mail	N	Cloze	Fill in entity	1.4M	93K CNN, 220K DM
Children's Book	N	Cloze	Fill in the word	688K	688K contexts, 108 books
SQuAD	N	Crowdsourced	Span of words	100K	536
MS MARCO	Y	User logs	Human generated	100K	1M passages, 200K+ doc.

Table 1: Comparison of some properties of existing datasets vs MS MARCO. MS MARCO is the only large dataet with open ended answers from real user queries

Datasets have played a significant role in making forward progress in difficult domains. The ImageNet dataset [10] is one of the best known for enabling advances in image classification and detection and inspired new classes of deep learning algorithms [22] [13] [15]. Reading comprehension and open

domain question answering is one of those domains existing systems still struggle to solve [31]. Here we summarize a couple of the previous approaches towards datasets for reading comprehension and open domain question answering.

One can find a reasonable amount of semi-synthetic reading comprehension and question answering datasets. Since these can be automatically generated they can be large enough to apply modern data intensive models. Hermann et al. created a corpus of cloze style questions from CNN / Daily News summaries [16] and Hill et al. has built the Children's Book Test [17]. Another popular question answering dataset involving reasoning is by Weston et al. [31]. One drawback with these sets is it does not capture the same question characteristics we find with questions people ask in the real world.

MCTest is a challenging dataset which contains 660 stories created by crowdworkers, 4 questions per story, and 4 answer choices per question [27], but real-world QA systems needs to go beyond multiple choice answers or selecting from known responses. WikiQA is another set which includes 3047 questions [32]. While other sets are synthetic or editor-generated questions WikiQA is constructed using a more natural process using actual query logs. It also includes questions for which there are no correct sentences which is an important component in any QA system like MS MARCO. Unfortunately, these sets are too small to try data demanding approaches like deep learning.

A more recently introduced reading comprehension dataset is the Stanford Question Answering Dataset (SQuAD) [26] which consists of 107785 question/answer pairs from 536 articles where the answer is span of paragraph. A few differences between MS MARCO and SQuAD is (a) SQuAD consisting of questions posed by crowdworkers while MS MARCO is sampled from the real world, (b) SQuAD is on a small set of high quality Wikipedia articles while MS MARCO is from a large set of real web documents, (c) MS MARCO includessome unanswerable queries and (d) SQuAD consists of spans while MS MARCO has human generated answers (if there is one).

3 The MS MARCO Dataset

In order to deliver true machine Reading Comprehension (RC), we start with QA as the initial problem to solve. Our introduction covered some of the key advantages of making very large RC or QA datasets freely available that contain only real-world questions and human crowdsourced answers versus artificially generated data. Given those advantages, our goal is that MS MARCO [5] - a large scale, real-world and human sourced QA dataset - will become a key vehicle to empower researchers to deliver many more AI breakthroughs in the future, just like ImageNet [10] enabled for image comprehension before.

Additionally, building an RC-oriented dataset helps us understand a contained yet complex RC problem while learning about all of the infrastructure pieces needed to build such a large one-million query set that helps the community make progress on state-of-the-art research problems. This task is also helping us experiment with natural language processing and deep learning models as well as to understand detailed characteristics of the very large training data required to deliver a true AI breakthrough in RC.

This first MS MARCO release contains 100,000 queries with answers to share the rich information and benchmarking capabilities it enables. Our first goal is to inspire the research community to try and solve reading comprehension by building great question answering and related models with the ability to carry out complex reasoning. We also aim to gather feedback and learn from the community towards completing the one-million query dataset in the near future.

This dataset has specific value-added features that distinguish itself from previous datasets freely available to researchers. The following factors describe the uniqueness of the MS MARCO dataset:

- All questions are *real, anonymized user queries* issued to the Bing search engine.
- The context passages, which answers are derived from, are extracted from *real Web documents* in the Bing Index.
- All of the answers to the queries are *human generated*.
- A subset of these queries has *multiple answers*.
- A subset of these queries have *no answers*.
- All queries are tagged with *segment* information.

The next sections outline the structure, building process and distribution of the MS MARCO dataset along with metrics needed to benchmark answer or passage synthesis and our initial experimentation results.

3.1 Dataset Structure and Building Process

The MS MARCO dataset structure is described in Table 2 below.

Field	Definition
Query	Question query real users issued to the Bing search engine.
Passages	Top 10 contextual passages extracted from public Web documents to answer the query above. They are presented in ranked order to human judges.
Document URLs	URLs for the top documents ranked for the query. These documents are the sources for the contextual passages.
Answer(s)	Synthesized answers from human judges for the query, automatically extracted passages and their corresponding public Web documents.
Segment	QA classification tag. E.g., tallest mountain in south america belongs to the ENTITY segment because the answer is an entity (Aconcagua).

Table 2: MS MARCO Dataset Composition

Starting with the real-world Bing user queries we filter them down to only those that are asking for a question (1) and the Web index documents mentioned in Table 2 as data sources, we automatically extracted context passages from those documents (2). Then, human judges selected relevant passages that helped them write natural language answers to each query in a concise way (3). Following detailed guidelines, judges used a Web-based user interface (UI) to complete this task (3 and 4). A simplified example of such a UI is shown in figure 2.

A feedback cycle and auditing process evaluated dataset quality regularly to ensure answers were accurate and followed the guidelines. In the back-end, we tagged queries with segment classification labels (5) to understand the resulting distribution and the type of data analysis, measurement and experiments this dataset would enable for researchers. Segment tags include

- NUMERIC
- ENTITY
- LOCATION
- PERSON
- DESCRIPTION (Phrase)

It is important to note that the question queries above are not artificially handcrafted questions based on Web documents but real user queries issued to Bing over the years. Humans are not always clear, concise or to the point when asking questions to a search engine. An example of a real question query issued to Bing is *{in what type of circulation does the oxygenated blood flow between the heart and the cells of the body?}*. Unlike previously available datasets, we believe these questions better represent actual human information seeking needs and are more complex to answer compared to artificially generated questions based on a set of documents.

To solve for these types of questions we need a system with human level reading comprehension and reasoning abilities. E.g., given a query such as *{will I qualify for osap if i'm new in canada}* as shown in figure 2 one of the relevant passages include:

You must be a 1. Canadian citizen, 2. Permanent Resident or 3. Protected person

A RC model needs to parse and understand that being new to a country is usually the opposite of citizen, permanent resident, etc. This is not a simple task to do in a general way. As part of our dataset quality control process, we noticed that even human judges had a hard time reaching this type of conclusions, especially for content belonging to areas they were not familiar with.

The MS MARCO dataset that we are publishing consists of four major components:

- *Queries*:These are a subset of user queries issued to a commercial search engine wherein the user is looking for a specific answer. This is in contrast to navigational intent which is another major chunk of user queries where the intent is to visit a destination website. The queries were selected through a classifier which was trained towards answer seeking intent of the query based on human labeled data. The query set was further pruned to only contain queries for which the human judges were able to generate an answer based on the passages that were provided to the judges.
- *Passages*: For each query, we also present a set of approximately 10 passages which might *potentially* have the answer to the query. These passages are extracted from relevant webpages. The passages were selected through a separate IR (information retrieval) based machine learned system.
- *Answers*: For each query, the data set also contain one or multiple answers that were generated by human judges. The judge task involved looking at the passages and synthesizing an answer using the content of the passages that best answers the given query.
- *Query type*: For each query, the dataset also contains the query intent type across five different categories – (a) description, (b) numeric, (c) entity, (d) person and (e) location. For example, "xbox one release date" will be labeled as $numeric$ while "how to cook a turkey" will be of type $description$.This classification is done using a machine learned classifier using human labeled training data. The features of the classifier included unigram/bigram features, brown clustering features, LDA cluster features, dependency parser features, amongst others. The classifier was a multi-class SVM classifier with an accuracy of 90.31% over test data.

Since the query set is coming from real user queries, not all queries explicitly contain "what", "where", "how" kind of keywords even though the intents are similar. For example, users could type in a query like "what is the age of barack obama" as "barack obama age". Table 3.1 lists the percentage of queries that explicitly contain the words "what", "where", etc.

Query contains	**Percentage of queries**
what	37.7%
how	15.2%
where	4.8%
when	2.2%
who	1.9%
why	1.4%
which	1.4%

Table 3: Percentage of queries containing question keywords

The following table shows the distribution of queries across different answer types as described earlier in this section.

Answer type	**Percentage of queries**
Description	49.3%
Numeric	31.2%
Entity	10.1%
Location	6.3%
Person	3.1%

Table 4: Distribution of queries based on answer-type classifier

4 Experimental Results

In this section, we present our results over a range of experiments designed to showcase characteristics of MS MARCO dataset. As we discussed in section 3, human judgments are being accumulated in

order to grow the dataset to the expected scale. Along the time line various snapshots of the dataset were taken and used in thoughtfully designed experiments for validation and insights. With dataset developing, the finalized experiment results may differ on the complete dataset, however, we expect observations and conclusions to be reasonably representative.

We group the queries in MS MARCO dataset into various categories based on their answer types, as described in subsection 3.1. The complexity of the answers varies greatly from category to category. For example, the answers to Yes/No questions are simply binary. The answers to entity questions can be a single entity name or phrase, such as the answer "Rome" for query "What is the capital of Italy". However, for other categories such as description queries, a longer textual answer is often required to answer to full extent, such as query "What is the agenda for Hollande's state visit to Washington?". These long textual answers may need to be derived through reasoning across multiple pieces of text. Since we impose no restrictions on the vocabulary used, different human editors often compose for the same query multiple reference answers with different expressions.

Therefore, in our experiments different evaluation metrics are used for different categories, building on metrics from our initial proposal [24]. As shown in subsection 4.1 and 4.2, we use accuracy and precision-recall to measure the quality of the numeric answers, and apply metrics like ROUGE-L [23] and phrasing-aware evaluation framework [24] for long textual answers. The phrasing-aware evaluation framework aims to deal with the diversity of natural language in evaluating long textual answers. The evaluation requires a large number of reference answers per question that are each curated by a different human editor, thus providing a natural way to estimate how diversely a group of individuals may phrase the answer to the same question. A family of pairwise similarity based metrics can used to incorporate consensus between different reference answers for evaluation. These metrics are simple modifications to metrics like BLEU [25] and METEOR [8], and are shown to achieve better correlation with human judgments. Accordingly as part of our experiments, a subset of MS MARCO where each query has multiple answers was used to evaluate model performance with both BLEU and pa-BLEU as metrics.

4.1 Generative Model Experiments

Recurrent Neural Networks (RNNs) are capable of predicting future elements from sequence prior. It is often used as a generative language model for various NLP tasks, such as machine translation [7], query answering [16], etc. In this QA experiment setup, we mainly target training and evaluation of such generative models which predict the human-generated answers given queries and/or contextual passages as model input.

- *Sequence-to-Sequence (Seq2Seq) Model*: Seq2Seq [30] model is one of the most commonly used RNN models. We trained a vanilla Seq2Seq model similar to the one described in [30] with query as source sequence and answer as target sequence.
- *Memory Networks Model*: End-to-End Memory Networks [29] was proposed for and has shown good performance in QA task for its ability of learning memory representation of contextual information. We adapted this model for generation by using summed memory representation as the initial state of a RNN decoder.
- *Discriminative Model*: For comparison we also trained a discriminative model to rank provided passages as a baseline. This is a variant of [20] where we use LSTM [19] in place of Multilayer Perceptron (MLP).

	Description	**ROUGE-L**
Best Passage	Best ROUGE-L of any passage	0.351
Passage Ranking	A DSSM-alike passage ranking model	0.177
Sequence to Sequence	Vanilla seq2seq model predicting answers from questions	0.089
Memory Network	Seq2seq model with MemNN for passages	0.119

Table 5: ROUGE-L of Different QA Models Tested against a Subset of MS MARCO

Table 5 shows the result quality from these models using ROUGE-L metric. While passages provided in MS MARCO generally contains useful information for given queries, the answer generation nature of the problem makes it relatively challenging for simple generative models to achieve great

	BLEU	pa-BLEU
Best Passage	0.359	0.453
Memory Network	0.340	0.341

Table 6: BLEU and pa-BLEU on a Multi-Answer Subset of MS MARCO

results. Model advancement from Seq2Seq to Memory Networks are captured by MS MARCO on ROUGE-L.

Additionally we evaluated Memory Networks model on an MS MARCO subset where queries have multiple answers. Table 6 shows answers quality of the model measured by BLEU and its pairwise variant pa-BLEU [24].

4.2 Cloze-Style Model Experiments

Cloze-style test is a representative and fundamental problem in machine reading comprehension. In this test, a model attempts to predict missing symbols in a partially given text sequence by reading context texts that potentially have helpful information. CNN and Daily Mail dataset is one of the most commonly used cloze-style QA dataset. Sizable progress has been made recently from various model proposals in participating cloze-style test competition on these datasets. In this section, we present the performance of two machine reading comprehension models using both CNN test dataset and a MS MARCO subset. The subset is filtered to numeric answer type category, to which cloze-style test is applicable.

- *Attention Sum Reader (AS Reader)*: AS Reader [21] is a simple model that uses attention to directly pick the answer from the context.
- *ReasoNet*: ReasoNet [28] also relies on attention, but is also a dynamic multi-turn model that attempts to exploit and reason over the relation among queries, contexts and answers.

	Accuracy	
	MS MARCO	CNN (test)
AS Reader	55.0	69.5
ReasoNet	58.9	74.7

Table 7: Accuracy of MRC Models on Numeric Segment of MS MARCO

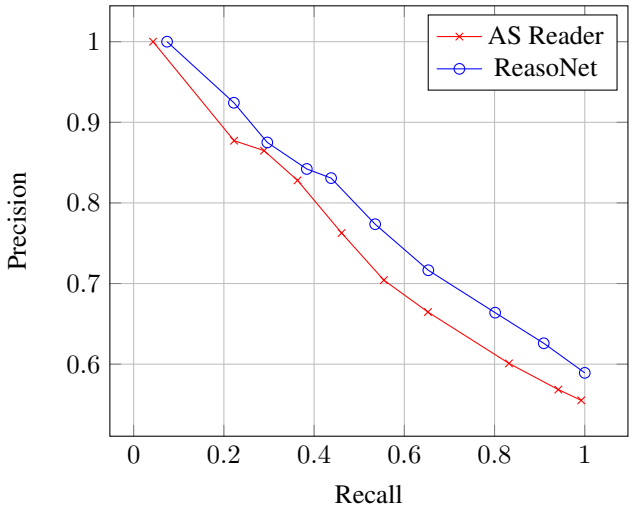


Figure 1: Precision-Recall of Machine Reading Comprehension Models on MS MARCO Subset of Numeric Category

We show model accuracy numbers on both datasets in table 7, and precision-recall curves on MS MARCO subset in figure 1.

5 Summary and Future Work

The MS MARCO dataset described in this paper above provides training data with question-answer pairs, where only a single answer text is provided via crowdsourcing. This simplicity makes the evaluation relatively easy. However, in the real world, multiple and equally valid answers are possible to a single question. This is akin to machine translation where multiple ways of translation are equally valid. Our immediate future work is to enrich the test set of the current dataset by providing multiple answers. We plan to add 1000 to 5000 such multiple answers in the dateset described in this paper.

Subsequent evaluation experiments on comparing single vs. multiple answers will be conducted to understand whether the model we have built has better resolution with multiple answers. The evaluation metric can be the same METEOR as described in the experiments reported earlier in this paper.

While MS MARCO has overcome a set of undesirable characteristics of the existing RC and QA datasets, notably the requirement that the answers to questions have to be restricted to an entity or a span from the existing reading text. Our longer-term goal is to be able to develop more advanced datasets to assess and facilitate research towards real, human-like reading comprehension. Currently, much of the successes of deep learning has been demonstrated in classification tasks [12]. Extending this success, the more complex reasoning process in many current deep-learning-based RC and QA methods has relied on multiple stages of memory networks with attention mechanisms and with close supervision information for classification. These artificial memory elements are far away from the human memory mechanism, and they derive their power mainly from the labeled data (single or multiple answers as labels) which guides the learning of network weights using a largely supervised learning paradigm. This is completely different from how human does reasoning. If we ask the current connectionist reasoning models trained on question-answer pairs to do another task such as recommendation or translation that are away from the intended classification task (i.e. answering questions expressed in a pre-fixed vocabulary), they will completely fail. Human cognitive reasoning would not fail in such cases. While recent work is moving towards this important direction [14], how to develop new deep learning methods towards human-like natural language understanding and reasoning, and how to design more advanced datasets to evaluate and facilitate this research is our longer-term goal.

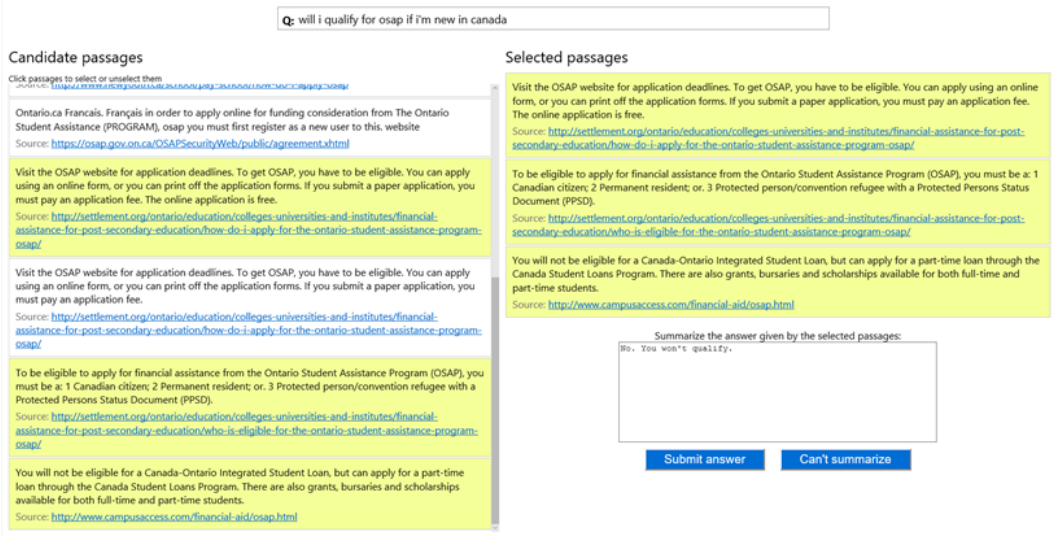


Figure 2: Simplified passage selection and answer summarization UI for human judges.

References

[1] Amazon alexa. `http://alexa.amazon.com/`.

[2] Amazon echo. `https://en.wikipedia.org/wiki/Amazon_Echo`.

[3] Cortana personal assistant. `http://www.microsoft.com/en-us/mobile/experiences/cortana/`.

[4] Google assistant. https://assistant.google.com/.

[5] Ms marco. http://www.msmarco.org/.

[6] Siri personal assistant. http://www.apple.com/ios/siri/.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[8] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72, 2005.

[9] G. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fe. Imagenet: Alarge-scalehierarchicalimagedatabas. *CVPR*, 2009.

[11] L. Deng and XD Huang. Challenges in adopting speech recognition. *Communications of the ACM*, 47(1):69–75, 2004.

[12] L. Deng and D. Yu. *Deep Learning: Methods and Applications*. NOW Publishers, New York, 2014.

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[14] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 2016.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

[16] Karl Moritz Hermann, Tomáš Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. 2015.

[17] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. 2015.

[18] G. Hinton, L. Deng, D. Yu, G. Dalh, and A. Mohamed. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM, 2013.

[21] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*, 2016.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[23] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

[24] Bhaskar Mitra, Grady Simon, Jianfeng Gao, Nick Craswell, and Li Deng. A proposal for evaluating answer distillation from web data.

[25] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[26] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. 2016.

[27] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. *EMNLP*, 2013.

[28] Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284*, 2016.

[29] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[30] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

[31] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merrienboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. 2015.

[32] Yi Yang, Wen tau Yih, and Christopher Meek. Wikiqa: A challenge dataset for open-domain question answering. *EMNLP*, 2015.

Accuracy and Interpretability Trade-offs in Machine Learning Applied to Safer Gambling

Sanjoy Sankar
Department of Computer Science and BetBuddy Ltd.
City, University of London London, UK
Sanjoy Sankar@city.ac.uk

Tillman Weyde, Artur d'Avila Garcez, Gregory Slabaugh
Department of Computer Science
City, University of London
a.garcez@city.ac.uk

Simo Dragicevic, Chris Percy
BetBuddy, London, UK
simo@bet-buddy.com, cwspercy@gmail.com

Abstract

Responsible gambling is an area of research and industry which seeks to understand the pathways to harm from gambling and implement programmes to reduce or prevent harm that gambling might cause. There is a growing body of research that has used gambling behavioural data to model and predict harmful gambling, and the industry is showing increasing interest in technologies that can help gambling operators to better predict harm and prevent it through appropriate interventions. However, industry surveys and feedback clearly indicate that in order to enable wider adoption of such data-driven methods, industry and policy makers require a greater understanding of how machine learning methods make these predictions.

In this paper, we make use of the TREPAN algorithm for extracting decision trees from Neural Networks and Random Forests. We present the first comparative evaluation of predictive performance and tree properties for extracted trees, which is also the first comparative evaluation of knowledge extraction for safer gambling. Results indicate that TREPAN extracts better performing trees than direct learning of decision trees from the data. Overall, trees extracted with TREPAN from different models offer a good compromise between prediction accuracy and interpretability. TREPAN can produce decision trees with extended tests rules of different forms, so that interpretability depends on multiple factors. We present detailed results and a discussion of the trade-offs with regard to performance and interpretability and use in the gambling industry.

1 Introduction

The application of machine learning to understand gambling pathways to harm and addiction is a new and growing field of study. Account-based gambling, whether via Internet or retail channels, whilst traditionally used for marketing purposes, has revolutionized this field of study due to the amount of data available to identify early warning signs of potentially harmful behaviour [7]. Such data was previously anonymous or unregistered, and not attributable to an individual player. However, the

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

quantity of data simultaneously opens up questions of how best to interpret the data and its results: specifically, how to transform raw gambling session data into meaningful, descriptive variables, called behavioural markers, and how to relate those descriptive variables to an individual who is potentially at risk of harm or addiction.

There are two important benefits of being able to predict potential harm in gambling behaviours. The first is improved player protection. By identifying individuals whose play pattern approximates those who have previously experienced harm, the gambling operator can choose to share information or advice with the player that may support healthy engagement with the gambling platform. Alternatively, the operator may choose to restrict marketing activity or platform activities for that player for a certain period of time. For this to happen effectively, interpretability of results is important. The second benefit are more stable, long-term revenue flows to gambling operators, since gamblers that might use their platform less intensively than before may do so with greater security and satisfaction.

Whilst the current machine learning methods offer good prediction performance, their effectiveness will be limited by the machine's inability to explain its decisions and actions to users. Explainable machine learning will be essential if users are to understand, appropriately trust, and effectively manage this incoming generation of artificially intelligent partners [8]. In the context of gambling, whilst machine learning algorithms have demonstrated early promise by predicting potentially harmful gamblers [10, 12], the industry uptake of such systems will primarily be dependent upon the regulators' and gambling operators' ability to understand and effectively use them. In an effort to overcome these challenges, previous research [15] has applied the TREPAN knowledge extraction method to neural networks in an effort to understand aspects of harmful gambling behaviour e.g. which kinds of profiles fit into problematic gambling, which attributes explain players who have such profiles? Such questions are motivated by industry insight from a responsible gambling conference in Vancouver (New Horizons in Responsible Gambling, 2016) [9] and an industry seminar in London (Responsible Gambling Algorithms Roundtable, 2016) [8], in which gambling operators, treatment providers and public policy officials set out the need for effective interpretation of such complex machine learning algorithms.

In this paper, we apply TREPAN to random forests and neural networks and offer the first comparative analysis of extracted trees from different models with different parameters regarding their accuracies and interpretability. This is also the first comparative study of knowledge extraction for safer gambling. Results indicate that TREPAN is a useful technique to aid interpretation of random forest and neural networks, leading to improved performance compared to standard decision trees. Different models, extraction parameters and tree types lead to varied loss of accuracy and degrees of interpretability.

The remainder of this paper is organised as follows: Section 2 discusses the related work in the application of machine learning to understand and interpret gambling behaviour. Section 3 describes the extraction and comparison methodology, including the changes to TREPAN to enable application to both random forests and neural networks. Section 4 presents the results comparing model accuracy and fidelity to the original random forest and neural network and between different types of extracted trees. Section 5 discusses the interpretability of our empirical results, and concludes the need for further research of understanding and measuring algorithm interpretation.

2 Related Work

2.1 Predicting Harm in Gambling

Machine learning algorithms have only recently been applied to this field of study as a way of predicting potentially harmful gambling [10, 12]. In [10], data obtained from the gambling operator International Game Technology PLC (IGT) was used to describe Internet gambling self-excluders in terms of their demographic and behavioural characteristics. Data analysis approaches and methods for improving the accuracy of predicting self-excluders are developed by hand towards inferred behaviour models. Supervised machine learning models were evaluated in [14] in the context of predicting which gamblers could be at risk of problem gambling. Their results suggest useful but general methods and techniques for building models that can predict gamblers at risk of harm.

Building on the work from the live action sports betting dataset available from the Division on Addiction public domain, in [12] nine supervised learning methods were assessed at identifying disordered Internet sports gamblers. The supervised learning methods include logistic regression and

other regularized general linear models (GLM), neural networks, support vector machines (SVM) and random forests. The results ranged from 62% to 67% with a random forest model reaching the highest prediction accuracy on a test set. A key finding from [10] was that the random forest technique performed the best in overall model accuracy (87%). The test set accuracy of the logistic regression model was the lowest (72%), with Bayesian networks in second and neural networks in between.

However, a major limitation of this study was the lack of interpretability of the models. The random forests were very difficult to interpret, consisting of 200 binary decision trees of unlimited depth. The neural network used was perceptron with a a single hidden layer, with 33 inputs, 17 hidden neurons and 2 outputs, one for self-excluding players and the other for the control group players. With more than 500 weights, the neural network was also a black box. The Bayesian network, which used the K2 algorithm, included hundreds of separately defined conditional probabilities and also failed to instigate useful insight about the problem when shown to industry experts.

2.2 Industry Need for Knowledge Extraction and Explainable Prediction Models

As reported in [15], we polled the audience at a related presentation at the 2016 New Horizons in Responsible Gambling conference to explore the importance of knowledge extraction and algorithm interpretability. Respondents were asked whether they would prefer a responsible gambling assessment algorithm that provided a 90% accurate assessment of problem gambling risk that they could not unpack or understand, or a model that provided a 75% accurate assessment that was fully interpretable and accountable. Only 20% chose the more accurate model, with 70% preferring to sacrifice 15 percentage points of accuracy for greater interpretability (10% were uncertain or felt it depended on the circumstances).

In a further exercise undertaken at the Responsible Gambling Algorithms roundtable event held in London in 2016 [8], we asked senior industry stakeholders for their views on the importance of understanding and interpreting algorithms. Senior executives and experts, including participants drawn from gambling operators as well as representatives from treatment providers and the UK Gambling Commission and the UK Responsible Gambling Strategy Board, were asked if accuracy of algorithms for recognising gamblers at risk was considered a second priority compared to the need for understanding them. The unanimous consensus from the participants expressed a preference for a more understandable algorithm over a more accurate one. For example, Dirk Hansen, CEO of GamCare, the UK's largest problem gambling treatment provider, stressed the value of interpretability, especially as this can be an advantage when providing treatment as the counsellor has specific and relevant behavioural indicators to discuss. From a regulatory perspective, Paul Hope, Director from the UK Gambling Commission, the industry regulator, stated that greater model understanding would be a higher priority compared with greater accuracy.

2.3 Knowledge Extraction from Neural Networks

This paper focuses on knowledge extraction by using random forests and artificial neural networks and TREPAN on a new IGT dataset to not only predict, but also describe, self-excluders through knowledge extraction. Previously, in [15] a variant of TREPAN was applied to the neural network model trained on gambling data in [10] to produce compact, human-readable logic rules efficiently. To the best of our knowledge, this was the first industrial-strength application of knowledge extraction from neural networks, which otherwise are black boxes and unable to provide the explanatory insights which are required in this area of application. The research demonstrated that through knowledge extraction one can explore and validate the kinds of behavioural and demographic profiles that best predict self-exclusion, while developing a machine learning approach with greater potential for adoption by industry and treatment providers. Experimental results in [15] reported that the rules extracted achieved high fidelity (87%) to the trained neural network while maintaining competitive accuracy (1 percentage point reduction in overall accuracy) and providing useful insight to domain experts in responsible gambling. This raises the real possibility of implementing algorithms for responsible gambling that offer both high accuracy and high transparency and interpretability. However, a limitation of this research is that it did not apply TREPAN to other machine learning methods, notably random forests, which were the most accurate model in [10].

3 Method

Gambling Data and Data Preparation The IGT dataset is based on gambling behavioural data made available by IGT collected from 4th December 2014 to 30th June 2016 from the regulated Internet gambling jurisdiction of Ontario, Canada. The sample data for the prediction model development and testing was based on Internet casino play and comprised 13,615 control group players and 449 self-excluders who self-excluded for at least six months. Self-exclusion is only a secondary indicator of disordered gambling behaviour, but specifically voluntarily exclusion from gambling platforms for significant periods of time has been previously used a dependent variable for developing models to predict potential harm in gambling [10].

The attributes of these players' raw activity data are a de-identified player unique ID, date of play, start time and end time of play sessions, type of game, game name, bet amount, and win amount. A number of behavioural markers are extracted that represent known aspects of risk, such as how much time gamblers spend on-line or how much they bet and how this evolves over time. For details of how the behavioural markers were generated, please see [10]. In addition to the 33 features used in [10], an additional 17 features were engineered and added to this data set to model additional behavioural markers around loss behaviours, such as increasing losses, increasing variation in the size of losses, and increasing loss chasing behaviours.

After processing the dataset contained 14,112 samples, each comprising 50 features per player. For each set, standard descriptive statistics measures are use to identify nature of data distribution and variance. A small number of samples with missing information was removed. This led to above mentioned of 13,615 control group player's samples and 449 self-excluder samples.

For the purpose of model building, a balanced training dataset was created by generating artificial samples for the minority class (self-excluder) using SMOTE [3].

Our approach for understanding gambling behaviour through machine learning is composed of three steps: gambling data preparation, models building using random forest and neural network algorithm, and knowledge extraction using TREPAN.

Model Development For the purpose of model development we a combination of random subsampling the control group and oversampling the self-excluders using SMOTE [3] to create a dataset with 1685 data points in each class.

The Random Forest models were built with forest size of 200 binary double trees, with unlimited depth. The neural network models contained 38 hidden nodes and 2 output nodes. A learning rate of 0.2 and momentum of 0.2 were used. The values of these hyper parameters were determined in a grid search. For comparison we also trained a standard decision tree with unlimited depth and number of leaves, splitting by the Gini criterion using the *scikit-learn* package in Python. The models are evaluated in 10-fold cross validation.

Knowledge extraction using TREPAN The original motivation of Craven's work in [4] was to represent a neural network model in a tree structure which could be more interpretable than a neural network classification model. This was in the context of a wider interest in knowledge extraction from neural networks [1,2], of which TREPAN has the advantage of being applicable to any oracle. In this work, the motivation was to apply Craven's method to a random forest model in addition to a neural network model and to explore the use of different types of decision rules in the generated trees. TREPAN generates decision rules of type *M of N* , *N of N*, *1 of N*, or *1 of 1*. In an *M of N* configuration, a tree node contains N distinct tests. If out of these N tests, M tests are satisfied, the tree will take one decision path, otherwise, the other path. *N of N* is a special case where $N = M$, creating a logical conjunction of tests. *1 of N* is another special case with $M = 1$, creating a logical disjunction of tests. *1 of 1* creates a standard decision tree with a single test per node. The type of rule is can be combined with different tree sizes in the tree generation process.

4 Experimental Results

Table 1 shows results of the random forest, neural network and decision tree models. In terms of classification accuracy, the results confirmed earlier research [10,15] that showed that random forests performed better than neural networks on this type of data and that both methods outperform a

Table 1: The structure and performance or random forest and neural network models.

Model Structure	Random Forest	Neural Network	Decision Tree
No of Trees	200		1
Tree Height	16 - 28		
Tree Leaves	343 - 432		
No of nodes in Hidden Layer		38	
Area under the ROC Curve	0.95	0.91	0.62
Accuracy	90%	84%	76%
True Positive Rate	85%	80%	80%
True Negative Rate	93%	88%	73%

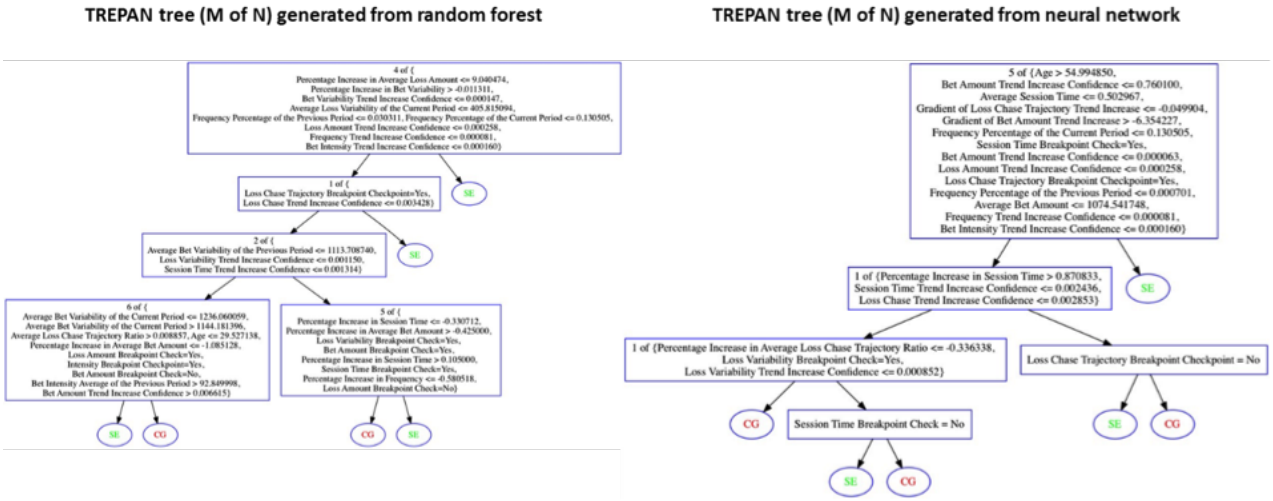


Figure 1: Decision trees generated with TREPAN for both a random forest and a neural network. The maximum number of internal nodes was set to 10.

standard decision tree with cross-validated accuracies of 90%, 84% and 76% respectively. We also tested neural networks with 2 and 3 hidden layers (with the same learning method as described above), but they did not improve results on this dataset.

Figure 1 depicts two trees generated by TREPAN.[1] CG at a leaf represents the control group class and SE represents the self-excluder class. For both trees, the maximum number of internal nodes is 10. It can be observed that considerable complexity is shown in the $MofN$ configuration the several internal nodes in each tree, especially for the larger trees.

Table 2 shows the performance of TREPAN generated trees for three different values of maximum number of internal nodes when applied to the two models. The fidelity of the model denotes the agreement between the TREPAN model and the original model. The accuracy of TREPAN model denotes the agreement with the training dataset used for model development. As expected, the overall accuracy of the TREPAN trees is reduced compared to the original models' performance for both models. The random forest was more accurate than the neural network, ranging from a difference of 4% to 1%, depending on the internal node size. The loss of accuracy is however lower for the trees extracted from the neural network (2–4%) then from the random forest (6–7%). The TREPAN trees extracted from the neural network showed higher fidelity to the original model (85%–87%), than those extracted from random forests (80%–81%). The internal node size of the TREPAN had little impact on accuracy or fidelity to the original models for random forests or neural networks.

The statistics of the tree and internal node structure that follow in Table 2 (M of N, feature counts, internal nodes, leaves, depth, feature count) show some differences of interpretability. The TREPAN trees where $max.internalnodes = 20$, when compared to those with $max.internalnodes = 5$, typically had more *M of N* nodes (5 and 4 v. 4 and 1 for random forests and neural networks), more

[1] Images generated with `http://www.graphviz.org/`.

Table 2: Descriptive summary of TREPAN decision trees generated for both random forest and neural networks.

Model Size	**Max. Int. Nodes = 5**		**Max. Int. Nodes = 10**		**Max. Int. Nodes = 20**	
Model Type	**RF**	**NN**	**RF**	**NN**	**RF**	**NN**
Accuracy	84%	80%	83%	82%	83%	80%
Fidelity	84%	87%	83%	87%	83%	85%
M of N Nodes	4	1	5	3	5	4
feature count in M of N	8,2,3,8	11	8,2,3,9,8	14,3,3	9,2,3,10,8	11,4,17,10
Internal Nodes	4	1	5	5	10	10
Leaves	5	2	6	6	11	11
Depth	4	1	4	4	7	8
Feature Count	21	11	30	22	37	48

Table 3: Comparison of results for both random forest and neural networks using four different TREPAN tree structures

Type	**Size of tree**	**Random Forest**		**Neural Network**	
		Accuracy	**Fidelity**	**Accuracy**	**Fidelity**
M of N	*Max internal node size = 5*	84%	84%	80%	87%
	Max internal node size = 10	83%	83%	82%	87%
	Max internal node size = 20	83%	83%	80%	85%
N of N	*Max internal node size = 5*	52%	52%	82%	78%
	Max internal node size = 10	69%	69%	79%	79%
	Max internal node size = 20	70%	70%	83%	81%
1 of N	*Max internal node size = 5*	51%	51%	83%	79%
	Max internal node size = 10	75%	75%	84%	79%
	Max internal node size = 20	78%	78%	85%	80%
1 of 1	*Max internal node size = 5*	51%	51%	47%	54%
	Max internal node size = 10	57%	57%	75%	70%
	Max internal node size = 20	68%	68%	75%	73%

leaves (11 and 11 v. 5 and 2), and more features used (37 and 48 v. 21 and 11). However, this more complex structure did not translate into a notable improvement performance.

In addition to the default $MofN$ type trees mentioned above with arbitrary M, we also generated TREPAN trees using $NofN$, $1ofN$, and $1of1$ structures to compare model performance, as described in table 3. The results show that for the random forest, the $NofN$ and $1ofN$ lost much accuracy compared to $MofN$, while there was no such loss for the neural network. This is an interesting result, as the $MofN$ tress were reported as hard to interpret in initial expert feedback, particularly with with large values of N and M. From a logical perspective, the $MofN$ nodes represents a much more complex test rule than $1ofN$ or $NofN$, which correspond to a logical rule with only N components in a disjunction or conjunction. The $1ofN$ case produced the best overall results for all tree types extracted from the neural network, including the single most accurate extracted tree (85%, for max. internal node = 20), which is event more accurate than the neural network model itself.

5 Discussion

The accuracy of random forest is throughout higher than that of the neural network, but irrespective of their performances, both the random forest and neural networks models are still black boxes to the end user. The decision tree generated directly form the data performs much worse than the more complex model, which confirmed the motivation of this study.

It was observed that increasing or reducing maximum number of internal nodes had little effect on the TREPAN model performance against the original model (random forest or neural network). This is an interesting result because the main motivation behind generating TREPAN model trees is to enhance the interpretation of a complex model via a simple logical rule representation. A tree with fewer nodes delivers generally a simpler tree structure with better interpretability. For example, our 200 tree

random forest model would represent 3,000 pages of A4, which makes is practically uninterpretable to humans. The smaller generated trees fit on a single page of A4, whilst reducing accuracy by a relatively small amount. The best extracted tree had 5 percentage points lower accuracy than the best model overall (random forest), which is well below the 15 point loss that was considered acceptable by 80% of respondents in return for full model transparency in [15].

However, as complex test rules are created with in the nodes, it becomes harder interpret models. With $MofN$ nodes, the accuracy of a TREPAN tree generated from a random forest model is always higher than that of the neural network model. This indicates that the TREPAN generated trees represent the structure of a neural network more accurately compared to a random forest model. A possible explanation for this that the operation of neural network nodes (summation and thresholding with a non-linear function) can be more easily represented with $MofN$ nodes. If M is between 1 and M there is a combinatorial explosion of possible test results that leads to different decisions, which makes the interpretation of a decision node a complex undertaking. However, for $NofN$ and $1ofN$ the neural network generated more accurate trees. This is an interesting result, as the $MofN$ tress were reported as hard to interpret in initial expert feedback, particularly with with large values of N and M. From a logical perspective, the $MofN$ nodes represents a much more complex test rule than $1ofN$ or $NofN$, which correspond to a logical rule with only N components in a disjunction or conjunction. This makes particularly the use of $1ofN$ trees generated from a neural network attractive, which are highly accurate, robust against tree size, and easier to interpret.

Future research should be aimed at extending the TREPAN algorithm to find new methods to optimise the tree structures to aid interpretability. This entails further research assessing the value and boundary conditions of the two parameters M and N for human interpretability.

From a user perspective, the internal structure of nodes and the shape of a tree are factors in addition to the size of the tree and number of features that influence ability to interpret or comprehend a decision tree model generated. We hypothesise that that limits of working memory will introduce a non-linearity in the ease of interpretation depending on the number of nodes and the number and type of tests within a node. Also, the congruence of decision tests with familiar concepts is likely to influence interpretability [11] and the usefulness for practitioners such as therapists or in automatically generated personalised messages to the user. In addition to optimising the tree structure, improved visualizations may aid human interpretability further. A deeper understanding of the the interpretability will require further research. Based on the current results, a combination of constraints and selection by expert practitioners can be a practical solutions. Ultimately the interpretability vs. accuracy trade-off will be for key stakeholders, such as regulators, scientific community, and industry to decide upon.

6 Conclusions and Future Work

Responsible gambling can benefit from machine learning models to recognise potentially harmful gambling behaviour. The industry does however demand models that are interpretable for professionals and can provide information to affected users. To fulfil this demand we have conducted the first comparative study of different types of decision trees extracted from neural network and a random forest models with TREPAN for safer gambling.

For a complex machine learning model, TREPAN generates relatively decision trees that provide an approximation of the complex models and lend themselves to human interpretation. We evaluated the extraction of small decision trees from neural network and random forest models with different parametrisations with regard to different metrics of classification performance and with regard to the properties of the resulting trees.

The results show that the loss of accuracy can be kept relatively small (between 0 and 7 percentage points), even for small trees. The complexity of the trees generated by TREPAN depends on the number of nodes and the structure of the rules in the decision nodes, where considerable complexity can occur. Although random forests make the most accurate predictions, the best performing decision tree was generated from a neural network. This was also using a simpler form of rules than other high performing trees and seems therefore to offer the best trade-off in this study between accuracy and interpretability.

However, complex rules inside decision nodes mean that the tree size is not the only metric relevant to interpretability, and further empirical work is needed to understand better what determines interpretability for a user. From an industrial assessment perspective, we propose a further study to assess the interpretability of TREPAN trees by domain experts and potentially for end users which can in turn lead to improved knowledge extraction methods.

References

[1] R. Andrews, J. Diederich, and A. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge Based Systems, 8(6), 373-389, 1995

[2] Garcez, A. S. D., Broda, K., and Gabbay, D. (2001). Symbolic knowledge extraction from trained neural networks: A sound approach. Artificial Intelligence, 125(1-2), 155–207.

[3] Bowyer, K., Chawla, N., and Hall, L., Kegelmeyer P. (2002). SMOTE: Synthetic Minority Over sampling Technique. Journal Of Artificial Intelligence Research, 16, pages 321-357.

[4] Craven, M., and Shavlik, J. (1996). Extracting Tree-Structured Representations of Trained Networks. Advances in Neural Information Processing Systems, 37–45.

[5] DARPA call *Explainable AI*: http://www.darpa.mil/program/explainable-artificial-intelligence

[6] Franca, M. V. M., Zaverucha, G., and Garcez, A. S. D. (2014). Fast relational learning using bottom clause propositionalization with artificial neural networks. Machine Learning, 94(1), 81–104.

[7] Gainsbury, S., Wood, R. (2011). Internet gambling policy in critical comparative perspective: The effectiveness of existing regulatory frameworks. International Gambling Studies,11(3), 309-32.

[8] ROUNDTABLE: http://www.bet-buddy.com/media/1190/responsible-gambling-algorithms-roundtable-1-august-2016-final.pdf

[9] Percy, C. (2016). Can 'BlackBox' responsible gambling algorithms be understood by users? A real-world example. New Horizons in Responsible Gambling conference paper, Vancouver, February 2016.

[10] Percy, C., Franca, N., Dragičević, S., and Garcez, A. S. D. (2016). Predicting online gambling self-exclusion: an analysis of the performance of supervised machine learning models. International Gambling Studies. DOI:10.1080/14459795.2016.1151913.

[11] Besold, T., Muggleton, S., Schmid, U., Tamaddoni-Nezhad, A., and Zeller, C. How does Predicate Invention affect Human Comprehensibility?. In Proceedings of the 26th International Conference on Inductive Logic Programming (ILP 2016, Sept. 4th-6th, London). Springer, Accepted.

[12] Philander, K. S. (2013). Identifying high risk online gamblers: a comparison of data mining procedures. International Gambling Studies. DOI: 10.1080/14459795.2013.841721

[13] Karim, A. and Zhou, S. X-TREPAN: a multi class regression and adapted extraction of comprehensible decision tree in artificial neural networks. arXiv:1508.07551, Aug, 2015.

[14] T. Schellinck and T. Schrans (2011). Intelligent design: How to model gambler risk assessment by using loyalty tracking data. Journal of Gambling Issues: 26(1), 51-68.

[15] Chris Percy, Artur S. d'Avila Garcez, Simo Dragicevic, Manoel V. M. França, Greg G. Slabaugh, Tillman Weyde: The Need for Knowledge Extraction: Understanding Harmful Gambling Behavior with Neural Networks. ECAI 2016: 974-981